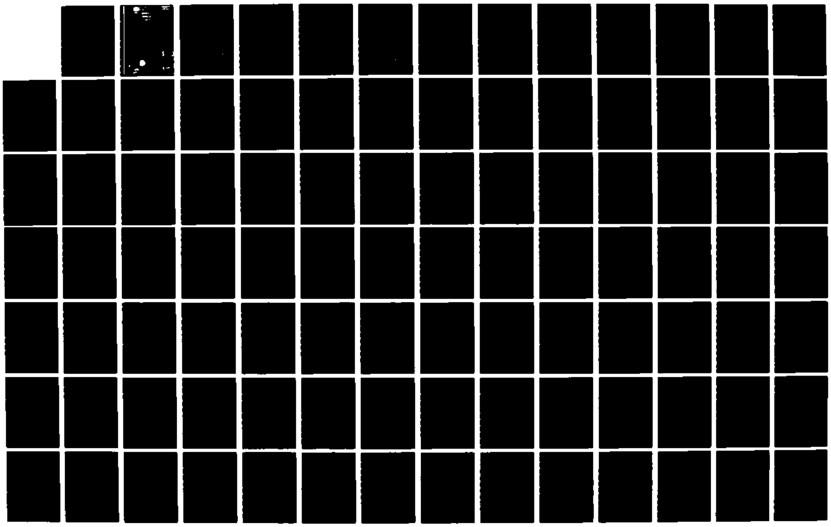
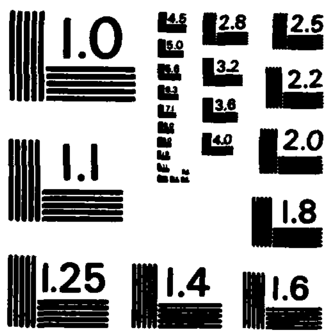
AD-A160 041 DYNAMIC MODELS OF DETAILED AND AGGREGATE PRODUCTION NETWORKS(U) CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER M MIZRACH SEP 85 ORC-85-10 1/2

UNCLASSIFIED N00014-76-C-0134 F/G 5/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

OPERATIONS RESEARCH CENTER

DYNAMIC MODELS OF DETAILED AND AGGREGATE PRODUCTION NETWORKS

by

Michael Mizrach*

ORC 85-10 September 1985

UNIVERSITY OF CALIFORNIA

BERKELEY

(12)

DYNAMIC MODELS OF DETAILED AND AGGREGATE PRODUCTION NETWORKS

by

Michael Mizrach*

ORC 85-10 September 1985

*Operations Research Center, University of California, Berkeley, California.

This research was supported by the Office of Naval Research under Contract N00014-76-C-0134 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ORC 85-10	2. GOVT ACCESSION NO. AD-A160041	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DYNAMIC MODELS OF DETAILED AND AGGREGATE PRODUCTION NETWORKS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael Mizrach		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0134
9. PERFORMING ORGANIZATION NAME AND ADDRESS Operations Research Center University of California Berkeley, CA 94720		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 337 015
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, Virginia 22217		12. REPORT DATE September 1985
		13. NUMBER OF PAGES 99
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Production Model	Detailed Networks
Processing Time	Aggregate Networks
Dynamic Activity Analysis	Production Planning
Scheduling Heuristics	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(See Abstract)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Dynamic Models of Detailed and Aggregate Production Networks.

by

Michael Mizrach

Ph.D

Industrial Engineering
& Operations Research

Sponsor:Office of Naval Research

[illegible]

Robert C. Leachman
Chairman of Committee

ABSTRACT

A decision support system for the management of a production organization must include an explicit model or models of the production system. When the organization faces a dynamic environment, dynamic models are most appropriate.

This ~~thesis~~ [illegible] presents dynamic models of production at two levels of details. First, ~~we~~ it extends existing activity analysis models for *detailed networks* to include the common situation of significant, finite *processing times* at each activity. This extension shows that dynamic activity analysis models require different indexing functions for the application of *labor and machine services* and for the application of *raw materials and intermediate products*. It is demonstrated that the improved accuracy of this model is very significant when the typical processing time in the system exceeds the planning time-grid intervals. The importance of ~~our~~ this development is in the identification of the underlying assumptions and range of appropriateness of the existing models and in the extension of these models to non-instantaneous production processes.

Additional Keywords: operations research; Scheduling heuristics

Many managerial decisions do not require a detailed description of the system. In the the second part of the thesis, we utilize the "parallel aggregation" approach developed by Leachman and Boysen, to reduce the representation of the network from the level of many (e.g. several thousands) detailed activities to a much smaller number of aggregates. We then use

dynamic linear activity analysis models; mathematical programming; industrial engineering

the structure of first model (which includes the effects of the *processing time*) to model the transfer relationships between the aggregates. (The parallel aggregation groups activities which belong to the same process).

The aggregate model that we propose is appropriate for both **Project** and **Manufacturing** Networks. To validate the models we compare results of simulations for the same networks as represented by a *detailed* model and an *aggregate* one. The results show that the aggregate model can be very useful to support managerial decisions in the evaluation of **Due-Date Feasibility** and **Capacity Planning**.

DEDICATION

To my wife Ayelet and to my children Inbal, Ofer and Anat.

ACKNOWLEDGEMENTS

It is my pleasure to thank my thesis advisor, Professor Leachman, for his support and for his enlighting suggestions which helped me a great deal in developing ideas and observations into mature and concise models.

I am also greatful to my Thesis-Committee members, Professor Glassey and Professor Shogan for their helpful comments and suggestions.

Table of Contents

Table of Exhibits

1. Introduction.

The modern era of the industrial society has brought an increasing importance to the area of Production Planning and Control . Many industries require high-technology processes which involve scarce and expensive equipment. In order to maintain profitability, the utilization of these resources must be planned and monitored . In other industries, the relatively cheap initial investment creates a very competitive market, so that efficient use of resources is a must for the survival of the firms . Today's complex production systems require sophisticated managerial decision-making techniques supported by computerized aids . Moreover, the decision-making process requires some *explicit or implicit model of the production system* .

Following Holt et al [1960] , we shall distinguish between three levels of managerial decisions :

a. **Strategic** decisions , such as selecting the line of products , selecting the size and location of plants and distribution centers , etc. We shall be interested in the question of project (or tasks) selection.

b. **Tactical** decisions . We shall be interested in two aspects at this level :

b.1 Determining capacity requirements, given the set of projects to be performed and their due-dates.

b.2 Capacity allocation among the various projects.

c. **Operational** decisions, such as detailed scheduling and dispatching.

The first part of our research (chapter 2) deals with the modeling of detailed production and project networks . We show that the Dynamic Linear Activity Analysis model (DLAAM), due to Shephard et al [1977] and Leachman [1982,1983] needs to be extended to include the aspect of *processing time.* As we show, this aspect can be very significant for accurate modeling of production. Our extended model will be denoted by the acronym DLAAMP. We further show that DLAAMP reduces to DLAAM only under some specific conditions concerning processing time.

Modeling of practical networks, especially for large projects or complex production systems, is cumbersome and very often results in scheduling formulations which exceed the capacity of today's computers. Even if we consider a Linear Program , which is the most efficient among the applicable procedures of mathematical programming, both the number of variables as well as the number of constraints are roughly *proportional* to $N \cdot T$, where N represents the number of "operations" that need to be done and T is the time-horizon . If we consider, for example, a time horizon of 50 periods ($T=50$) , N is practically limited to a few hundreds . Other techniques such as Dynamic Programming or Implicit Enumeration techniques are even more limited and solvable problems might be limited to a few tens of activities . (see Cooper [1976], Davis [1973], Pritsker et al. [1969], Talbot and Patterson [1978] and more).

This computational problem does not affect operational decision-making very often. The relevant time horizon for this level is usually short and often the detailed allocation of the resources, given the capacity levels which were decided at a higher hierarchical level, may be determined through the use of heuristical simulation programs. For example, we would like to draw the attention of the interested reader to a novel simulation algorithm for project scheduling with varying intensities which was developed recently by Dincerler [1984] .

On the other hand, the strategic and tactical decisions usually involve long-term time horizons, requiring different modeling and solution techniques . The reader should note that these levels of management are rarely interested in detailed production schedules *per se*. Instead, they are interested in *aggregate measures* such as the interaction between allocation of the capacities and completion dates of the various production tasks (i.e., projects) , or the capacities required to complete a set of projects with given due-dates. Furthermore, management is very often interested in performing several iterations ("what if ? " analysis) in order to consider several scenarios. Typically, the proposed solution to the needs of higher management levels is to represent the "oversized" detailed schedule by a network of *aggregate* activities with a much smaller number of *representing activities* .

We discuss the general concepts of aggregation and survey several aggregation approaches in chapter 3. In this chapter we explain why we prefer the approach of **parallel aggregation** as the appropriate one for general modeling purposes . The principles of "structural" aggregation are introduced in chapter 4 . Chapter 5 is devoted to a detailed discussion of models of parallel aggregation of detailed networks. The validation of our aggregate model is discussed in chapter 6 .

The contribution of this thesis is primarily in the following areas :

1. Extending Dynamic Linear Activity Analysis models for detailed networks to include *the important effect of processing time* .

2. Proposing a framework for parallel aggregation of production networks. We shall consider several types of production networks. We shall primarily deal with Project and Processing networks. Additional types such as Assembly-networks will be presented as extensions of the "pure" Project and Processing networks. (For a comprehensive discussion concerning the properties of these networks see section 2.1).

2. Modeling of Detailed Networks

2.1 Problem Definition

In this chapter we shall develop several models which attempt to represent dynamic production networks. These models will vary according to their view of the production system, gradually evolving to "catch" more realistic properties of actual production networks.

Production networks :

Our investigation will focus on multi-stage production systems which will be described by networks, where the nodes represent the activities and the arcs represent the transfer of intermediate products between these activities.

The exogenous inputs to the system are raw materials, labor of various types, energy, etc. The output of the system is a desired set of products. A processing-unit or output-unit of an activity will be referred to as a *transfer unit.*

Activities :

The basic elements of the network are the activities. The activities can be viewed as fixed technological processes applied to the materials entering the activity, converting it into "output", while consuming exogenous resources. This definition is natural for "processing" activities. In project-type activities we shall reinterpret the event of completion in terms of physical transfer units, as discussed below. We shall be interested in the managerial aspects of the transformation (use of materials, resources, etc). Our scope will exclude the pure technological aspects such as temperatures, pressures, cutting speed, etc. Through our discussion we shall distinguish between two types of activities which correspond to the two types of networks discussed in chapter 1 .

Project Activity - A distinct intermediate product is produced for each follow-on activity. That is, project activities have *vector* output. These outputs are simultaneously produced in proportion. Transfers of the various distinct products of an activity are simultaneous and must be unitary. All intermediate products are required before an activity can start. Each intermediate product has a unique source. A network consisting entirely of project activities is called a *Project Network.*

Processing Activity - The activity produces scalar output which may be allocated to various, alternative, follow-on activities. Transfer may or may not be integral depending on the nature of the product. The activity has at most one type of intermediate input (which may have alternative sources). A network consisting entirely of processing activities is called *Processing Network.*

Since each activity performs a fixed technological process we shall attempt to define explicit production functions to relate the inputs and the outputs of each activity.

Production Functions :

The equations which describe the input-output relationships of each activity are called "production functions". This term was first used in the field of Economics to determine the sets of *efficient* vectors of output which can be produced, given the *allocation* of resources. For further discussion see Koopmans [1951], Leontief [1951], and Morgenstern [1954].

The original definition of the production function fits instantaneous processes . When the process is evolutionary, we shall use an extended definition for the production function and assume "efficient" utilization of resources. For modeling purposes we shall focus on the rate of *applied* input as described by the function $y(t)$ and the corresponding output of the activity. In most cases we shall be interested in the cumulative production function $\Phi(Y)$, where $Y(t) = \int_0^t y(s) \cdot ds$ and $\Phi(Y)$ is the time function of cumulative outputs. For a detailed axiomatic discussion of the properties of Dynamic Production Functions see Hackman [1983].

Intermediate Transfers :

The outputs of the activities may serve as final or intermediate products or both. It will be useful to distinguish between two types of transfer policies :

Continuous transfers : Transfers are assumed to be completely divisible. Examples for this type of output are many Chemical processes. This transfer "policy" frequently serves as an approximation for discrete transfer systems.

Discrete transfers : Transfer occur only upon completion of a predetermined *integral* number of units. For a Project activity, the cumulative output of each product of the activity is defined as "1" upon completion of the task of the activity. For a Processing activity, one unit of output corresponds to a transfer batch consisting of a predetermined number of physical units of the intermediate product and output is transferred only in integral multiples of this unit.

2.2 Domain of Models :

Our discussion hereafter will be restricted to linear models. This domain of models implies the following simplifications :

a. Linear constraints : The constraints are represented by linear functions of the decision variables . In some models the variables will be treated as continuous, while others might include integer variables.

b. Proportional production functions : The resources (types of labor) are applied with fixed coefficients which relate consumption of resources to the "intensity" of the activity . Models evolving from the traditional "Dynamic Linear Activity Analysis model" will assume that consumption of materials and output are also proportional to the rate of the process . For the models assuming finite duration of processing we shall show that although the consumption rates of service-type resources are proportional among themselves and the consumption rates of raw materials (including intermediate products) are also proportional among themselves, the two vectors are not necessarily proportional to each other . Moreover, we shall show that, in general, *output* may be proportional to only one of the two rates at a particular point of time. We shall refer to this type of relationship as "semi-proportional production function".

The assumption of proportional or semi-proportional production functions has a restricting effect. Nonlinear effects such as set-ups , economies of scale and learning curves are not represented correctly and might require readjustment of the coefficients for different sizes of production batches, for an appropriate linear approximation .

2.3 Dynamic Linear Activity Analysis Models

The models introduced in this section were first introduced by Shepard, Al-Ayat and Leachman [1977] and further developed by Leachman [1982,1983]. However, simplified versions of these models are found in most linear programming formulations of production planning problems. We shall use the acronym DLAAM to refer to these models.

These models assume (explicitly) the following :

a. Linear objective functions, linear constraints and proportional production functions.

b. Directed production network

c. Time is divided into fixed intervals, called periods. Period t denotes the interval $(t-1,t]$. Our conventions will be that applications of resources are constant during the intervals, and that intermediate products which are applied as inputs in period t must have been produced before this period.

For the sake of clarity we shall add the following assumptions which will enable us to concentrate on the essentials of the models :

d. Resources are completely divisible (allocation of resources matches the actual application).

e. Transfer time of intermediate products between activities is negligible. (Obviates the need for "book keeping" of fixed lags). As a logical consequence we shall apply the simplifying assumption that no intermediate-products are kept "in front" of the activities.

f. Scalar outputs : Each activity produces one type of output. These outputs can be either intermediate products or final outputs. Final outputs are regarded as intermediate products delivered to "sink" activities.

g. Scalar inputs : Each activity uses one type (or alternative types) of intermediate product input. This input can come from various sources within the system. (A slightly different formulation can be used to model networks with complementary inputs, e.g. assembly).

The models are developed below for activities whose transfers are of the "processing" type. The "project" type systems can be regarded as a simpler case and are treated easily by the discrete version of the DLAAM . As mentioned above, our attention will be focused on the transfer relationships among the activities.

Notation :

a. Variables :

$z_i(t)$	"intensity" of activity i, defined in this case as the number of transfer units produced by activity i during interval t
$Z_i(t)$	cumulative intensity of activity i, defined as the cumulative number of transfer units produced by activity i till the end of period t .
$V_{ij}(t)$	cumulative number of units, transferred from activity i to its follower j up through time t .
$I_i(t)$	inventory of transfer units produced by activity i and not allocated to the followers at time t.
$Z_i^P(t)$	the cumulative number of intermediate products consumed by time t, also referred to as the cumulative "starts" of activity i.

b. Parameters :

a_{ki}	amount of resource k required by activity i per unit intensity.
$\bar{a}_{ij}$	number of units produced by i , required per unit intensity of j .
$x_k(t)$	capacity of resource k at time t (work-hours per hour) .

A visual description of the system is illustrated in figure 2.3-1 .

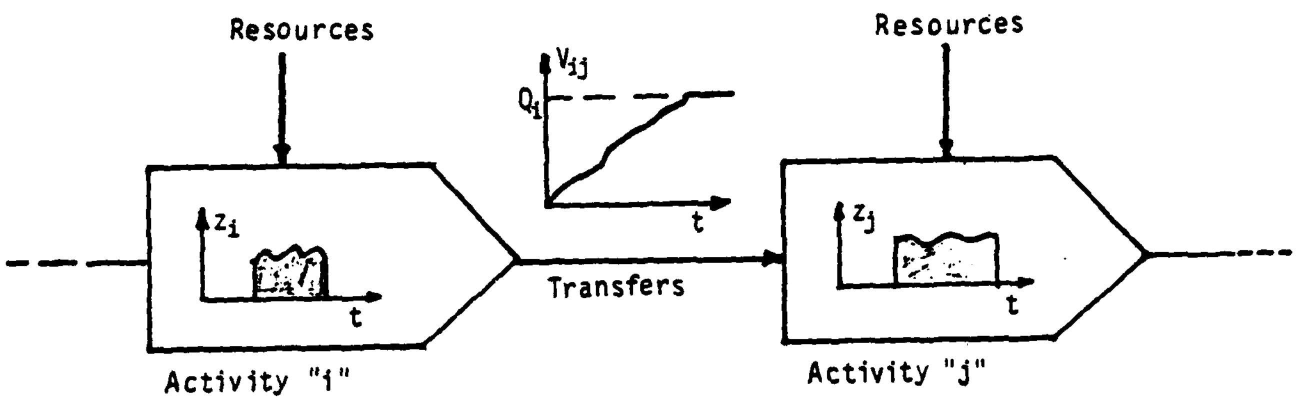


Figure 2.3-1: An illustration of a production system

2.3.1 DLAAM with Continuous (Completely Divisible) Transfers .

Under DLAAM, the technological transformation carried out by an activity is described by a proportional production function, characterized by the technological coefficients $\{a_{ki}\}$ and $\{\bar{a}_{ij}\}$.

The intermediate products variables, $V_{ij}(t)$ are non-negative and completely divisible . The resulting model is :

1) Inventory balance for the predecessor :

$$\sum_j V_{ij}(t) \le Z_i(t) + I_i(0)$$

2) Inventory balance for the follower :

$$\bar{a}_{ij} Z_j(t) \le V_{ij}(t-1)$$

The displacement of one time unit is due to the convention that there are no transfers within a time-interval .

3) Capacity balance :

$$\sum_{i=1}^{N} a_{ki} z_i(t) \le x_k(t)$$

4) Upper bound on the intensity of the activity :

$$z_i(t) \le z_i^{\max}$$

$z_i^{\max}$ can result from two reasons :

a. A resource which is not modeled explicitly in the production function but imposes an upper bound on the rate of the activity such as working space , electric power , budget constraints ,etc.

b. Technological bound on the rate of the process : maximal RPM for metal machining , processing time in film developing ,etc.

5) Non-negativity :

$z_i(t) \geq 0,\ I_i(t) \geq 0,\ V_{ij}(t) \geq 0$

An important observation is that we can relate the maximal cumulative intensity of the follower to the actual cumulative intensity of the predecessor by merging the first two constraints :

$$\sum_j \bar{a}_{ij} Z_j(t) \leq Z_i(t-1) + I_i(0)$$

2.3.2 DLAAM with Discrete Transfers :

Few production systems are characterized by continuous transfers . Continuous transfers are often used to approximate production systems with discrete transfers. A DLAAM model of a discrete transfer system is very similar to the continuous transfers model and differs only in the domain restrictions of the transfer related variables $V_{ij}(t)$ which are restricted to be non-negative integers.

2.4 Models Assuming "Finite Processing Time" .

So far we have regarded the technological process of the activity only through the technological coefficients which relate outputs to inputs. Another dimension which was overlooked in previous models is the duration of the process . When we observe most technological processes, continuous or discrete, we find that there is a minimal duration required to process a transfer unit . The importance of this phenomena will be demonstrated by the following simple example :

Let's assume the following values for the state variables of a continuous transfers system which consists of two activities, a predecessor i and a follower j :

$I_i(0) = 0$; $Z_i(5) = 4$

According to the DLAAM model, at period 6, activity j can start working on 4 units which could be delivered from i . Now, let's add the information that it takes at least 6 periods to process a unit in activity i . The immediate conclusion is that though the amount of work invested at activity i by period 5 is *equivalent* to the amount required to finish 4 units, this work-content must have been applied to more than 4 units since no unit is ready at the end of period 5.

2.4.1 Illustrative Model .

In this section we shall introduce an approximate model which illustrates the effect of processing-time . This model gives us an interesting insight about the functional relationship between the cumulative intensities of activities i and j . We shall observe that the introduction of processing-time makes a significant change in this relation, compared with the previous models . This insight will lead to the development of an appropriate detailed planning model in the next section. The illustrative model requires two additional assumptions:

a. Each activity i has a predetermined task, to produce Q_i transfer units which are delivered to its followers. This assumption is required only for the sake of this illustrative model and will not apply to the detailed models.

b. The processing of each transfer unit cannot be interrupted by another unit (each "server" will finish the unit it is processing before starting another one). The term "server" will be used as a generalization to any combination of labor and machine services required to perform the technological process . A transfer unit is a group of (identical or different) items which is processed simultaneously by a server. In most cases the transfer unit is a single item. The implication of this assumption is that we would rather complete one unit and deliver it "upstream" in the system, than produces two halves of two units. This assumption is quite reasonable in most production environments.

In order to relate the pair of activities i and j, we shall use normalized notations where the intensity $z_i(t)$ will be defined as the fraction consumed during interval t of the total amount of resources required by activity i to complete Q_i units . Based on this definition of z , we shall further assume that resources can be applied in the production of a transfer unit at a maximal rate of $\frac{1}{D_i \cdot Q_i}$ where D_i is the minimal processing time of a transfer unit through activity i .

Let's observe a typical activity which has a minimal processing time D_i for each transfer unit. For the sake of clarity we illustrate the properties of this activity assuming a constant rate of "starts" (the time at which a server starts processing a transfer unit). Furthermore, we shall assume that each server operates at its maximal rate, such that each unit stays D_i time units in process. In the "earliest" case the "starts" correspond with the rate of the inflow. The description of the intensity in this model is only an approximation and will be replaced by a more accurate formulation in the next section.

Computational comment: In order to avoid some "edge" effects which distort the linear relationships that we show, we shall assume that Q is large enough such that the system can be approximated by continuous transfers.

The corresponding relationships are illustrated in figure 2.4.1-1

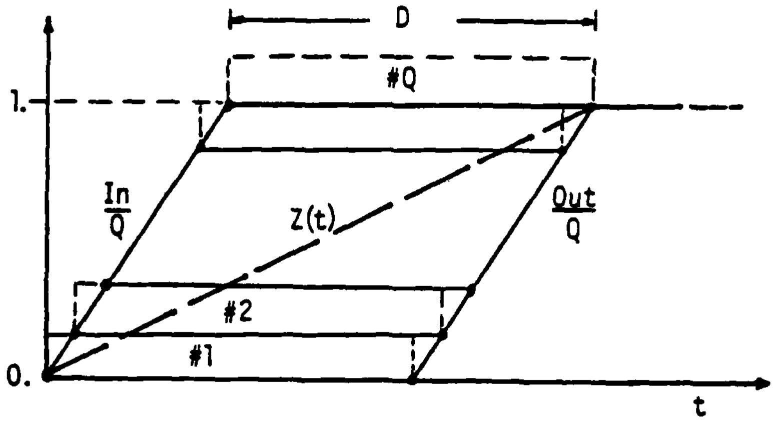


Figure 2.4.1-1: Input-output relationship of an activity with finite processing-time.

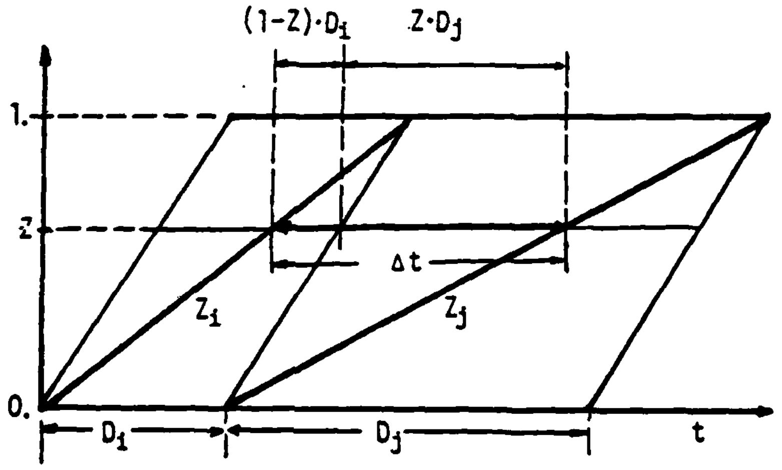


Figure 2.4.1-2: Transfer relationships between two activities with finite processing-time.

We would like to draw the attention of the reader to the following observations:

a. No output was released till time D_i, though the activity accomplished approximately 65% of its processing task at that time. (Measured by the cumulative use of resources : $Z_i(t)$).

b. The rate at which resources are applied is distinct from the rate of "starts". Different combinations of cumulative "starts" and/or different cumulative intensity ,will result in different curves of output.

c. The fastest (earliest) curve of the normalized output is the curve of the normalized input, delayed by D_i time units.

Figure 2.4.1-2 illustrates the transfer relationships between activities i and j . When the follower j operates at its earliest mode, the curve of its "starts" is identical with the curve of output of the predecessor i .

It is easy to see from figure 2.4.1-2 that there is a time-lag Δt between the cumulative normalized intensities of the pair. When activity i operates at its maximal intensity the time-lag is a linear function of the level of the cumulative intensity Z :

$$\Delta t(Z) = (1-Z)D_i + Z \cdot D_j$$

This time lag can be used to calculate an upper bound of the cumulative intensity $Z_j(t)$,given the "history" of $Z_i(t)$:

$$Z_j(t) \leq Z_i[t-(D_i + Z_j(t) \cdot (D_j - D_i))]$$

This nonlinear bound will not be used in future models.

In the next section we develop more rigorously the relationship between input and output of an activity with processing time.

2.4.2 Detailed Transfer Relationships :

The previous model illustrated several effects related with the "processing time" phenomena. As we shall show in the current model, the derivative of the cumulative intensity curve is not necessarily constant even when the derivative of the input curve is. The next model will cure this inaccuracy and provide a framework for practical modeling of Dynamic Production Networks.

Assumptions :

a) *General assumptions and conventions :*

1) The system is represented by a directed and acyclic network.

2) The usage of service type resources is proportional and can be indexed by a scalar intensity function.

3) If production by any activity requires a mix of intermediate products, this mix is fixed. We shall index the cumulative application of intermediate products by the cumulative starts function $Z_i^P(t)$.

4) Constraints and Objective Function are linear.

5) Intermediate products which are applied at period t must have been completed in previous periods.

b) *Assumptions specific to the model :*

6) Transfers are processed by the FIFO discipline.

7) It takes exactly D_i time units to process a transfer unit through activity i , for each i. (D_i can assume any positive number).

8) Each unit is processed by only one server. This server will process the unit with a uniform rate of $\frac{1}{D_i}$ units from start to finish. Though this assumption seems to be quite restrictive, it is acceptable in most production environments. The reader should consider the similarity to the assumptions of CPM networks.

Additional discussion of assumptions 6,7 and 8 accompanied by examples can be found in appendix B.

2.4.2.1 Starts occur only at the beginning of a period :

We shall develop the formulation for discrete transfers. The first model will assume that new starts occur only at the beginning of a period. This assumption will be relaxed later at the cost of a more complicated model.

Notation

As shown before, the amount of work which was invested in transfer units is not identical with the actual output of an activity with a finite processing time. For this type of activity we shall have to adjust our definitions to distinguish between the work and the output :

$z_i(t)$	The rate of application of service type resources, measured in processing units. (Each unit equals the amount of "work" which has to be invested in a single transfer unit).
$Z_i(t)$	The cumulative application of service type resources, denoted by cumulative intensity.
$U_i(t)$	the cumulative actual output of activity i by time t.

Formulation :

1. Bounds on the intensity of activity j at time t :

1.1 Available intermediate products :

The number of processable units at period t is the difference between the number of units started by activity j by the end of period t, and the number of units which were completed by time $t-1$. During an interval, each unit is processed with a maximal rate of $\frac{1}{D_j}$ for at most $\min(1,D_j)$ time units. Thus the (dynamic) maximal rate is :

$$z_j(t) \le [Z_j^P(t) - U_j(t-1)] \cdot \frac{min(1,D_j)}{D_j}$$

1.2 Application upper limit :

At any time, activity j can not use, simultaneously, more than the maximum number of servers, denoted by n_j.

$$z_j(t) \leq \frac{n_j \cdot min(1,D_j)}{D_j}$$

Comment : n_j is not necessarily integer. (e.g. the volume of a tank in a chemical process).

1.3 Remaining work :

The difference between the total work to be performed on the units which were started by the end of period t , and the amount of work which was done so far , is the maximal additional work which can be done during period t :

$$z_j(t) \leq Z_j^p(t) - Z_j(t-1)$$

2. Production function : (maximal output, given the application of inputs.)

A unit started at the beginning of a period would be available for transfer, $\lceil D_j \rceil - 1$ time periods afterwards. (Under assumption 8), thus :

$$U_j(t) = Z_j^p(t+1 - \lceil D_j \rceil)$$

3. Cumulative starts at activity j by the end of period t :

Under assumption 8 and the assumption that new starts occur only at the beginning of a period, the number of new starts can be determined as follows :

The number of units which remain from starts in previous periods is $Z_j^p(t-1) - U_j(t-1)$. The amount of work which was invested during period t equals $z_j(t)$. Since each new unit occupies the server for the maximum of its "duration" and a time unit, the number of new starts is :

$$Z_j^P(t) = \max \begin{cases} \lceil z_j(t) \cdot max(1,D_j) \rceil + U_j(t-1) \\ Z_j^P(t-1) \end{cases}$$

To understand this result, it might be helpful to subtract $Z_j^P(t-1)$ from both sides. The upper equation gives the difference between the number of units on which work was performed during period t, and the number of "old" unfinished starts.

A close examination of the last equation reveals that constraint 1.1 is included in this equation and thus can be removed. We can summarize the resulting model as follows :

1) Intensity :

$$z_j(t) \le \begin{cases} n_j \cdot \dfrac{min(1,D_j)}{D_j} \\ Z_i^P(t) - Z_j(t-1) \end{cases}$$

2) Actual output :

$$U_j(t) = Z_j^P(t+1 - \lceil D_j \rceil)$$

3) Number of starts :

$$Z_j^P(t) = \max \begin{cases} \lceil z_j(t) \cdot max(1,D_j) \rceil + U_j(t-1) \\ Z_j^P(t-1) \end{cases}$$

4.1) Upper bound on starts for complementary inputs :

$$Z_j^P(t) \le \min_i \lfloor V_{ij}(t-1)/\bar{a}_{ij} \rfloor \quad : \bar{a}_{ij} \neq 0$$

4.2) Upper bound on starts for additive inputs :

$$Z_j^P(t) \le \lfloor \sum_i V_{ij}(t-1)/\bar{a}_{ij} \rfloor \quad : \bar{a}_{ij} \neq 0$$

5) Capacity :

$$\sum_i a_{ki} z_i(t) \leq x_k(t)$$

6) Sum of transfers :

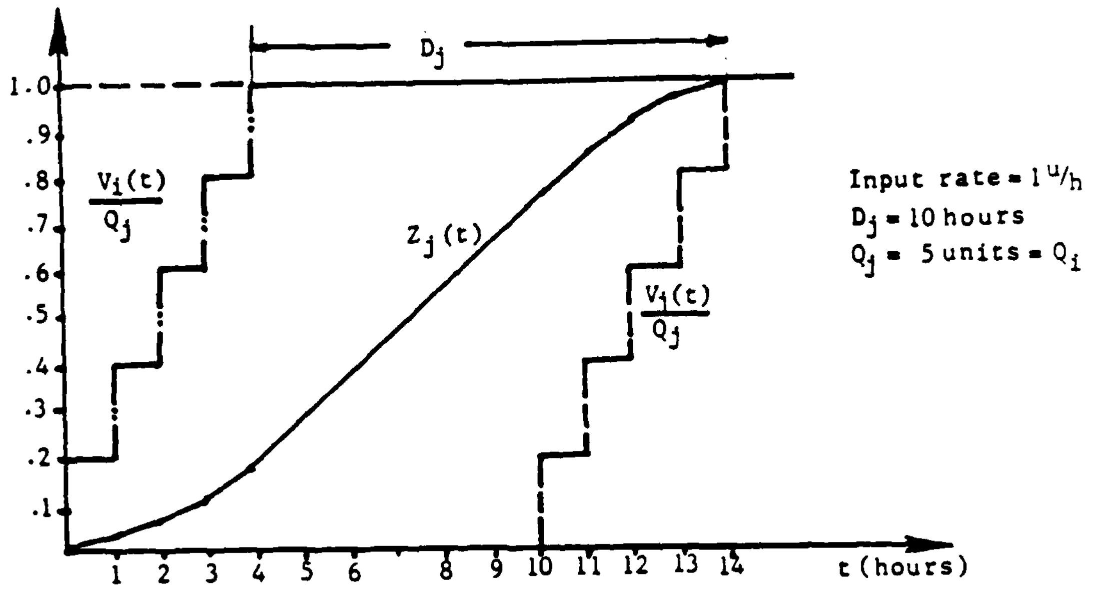
$$\sum_j V_{ij}(t) \leq U_i(t)$$

7) Non-negativity

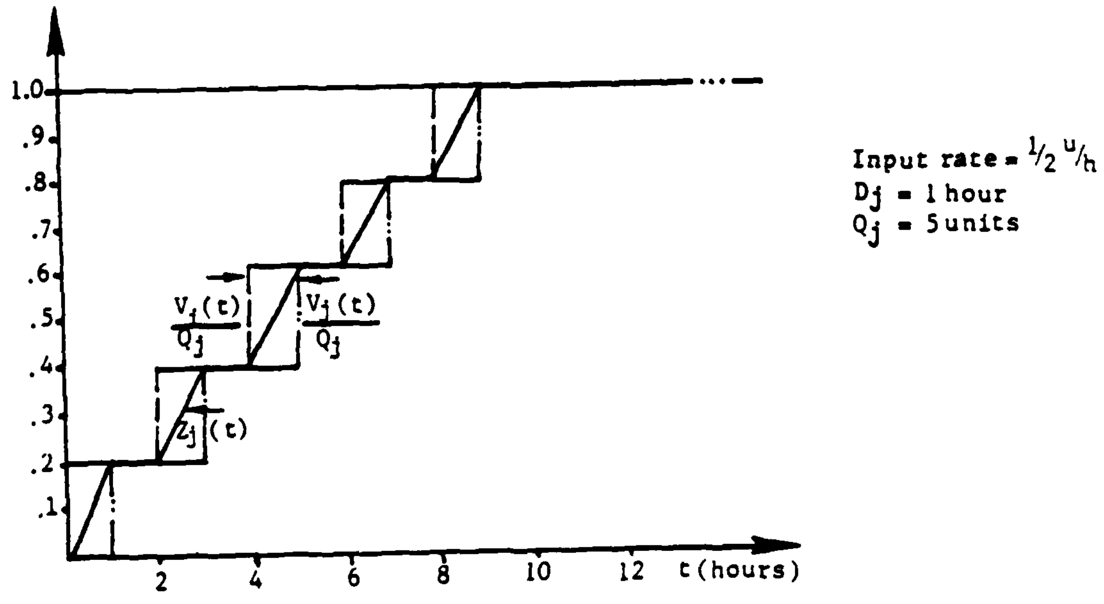
A description of the model is illustrated in fig 2.4.2 for two cases :

a) The rate of input is larger then $\frac{1}{D_j}$.

b) The rate of input is smaller then $\frac{1}{D_j}$.



a) rate of input is not bounding



b) rate of inout bounds the rate of orocessino

Fiqure 2.4.2: Detailed (uncapacitated) inout-outout relationshin of DLAAMP

Adaptation of the model for continuous transfers :

The models for continuous tra~sfers and for discrete transfers are similar except for the removal of some domain restrictions, i.e., the "rounding" functions in constraints 3 and 4 are removed.

2.4.2.2 The case of Intra-Period Starts

The previous assumption that new starts occur only at the beginning of a period is appropriate when D >> 1 . If we want the contribution of this assumption to the idle time at activity i to be less then, say, k % , we should choose the length of the period such that :

$$\frac{\lceil D_i \rceil - D_i}{\lceil D_i \rceil} \le \frac{k}{100}$$

In some cases this restri~tion will cause a very fine grid which makes the formulation impractical. We shall now propose an algorithm to relax this restriction by estimating the number of new starts in a period, when the starting of a new unit is not restricted to the beginning of the period.

Let us assume, for the moment, that the amount of work which was invested by activity i during period t, in new starts, equals $w_i(t)$. (This amount is measured in units of equivalent transfers.)

The actual number of starts which used $w_i(t)$ depends on the arrangement of the starts in previous periods on the various servers. We shall propose a conservative estimate which is a *lower bound* on the number of new starts :

Under assumption 8, the rate at which a unit is processed at activity i is $\frac{1}{D_i}$.

Let l_t^k denote the length of time at which (new start) unit k was processed during period t , then, by definition :

$$w_i(t) = \frac{1}{D_i} \cdot \sum_k l_i^k$$

It is also clear that $l_i^k \leq min(1,D_i)$ for each k. If we replace l_i^k by $min(1,D_i)$, for each k, we get a lower bound on the number of new starts, denoted by $\hat{k}(t)$:

$$\hat{k}(t) = \lceil \frac{w_i(t) \cdot D_i}{min(1,D_i)} \rceil$$

The reader should note that if all servers are empty at the beginning of the period, then $z_i(t) = w_i(t)$.

We can compute the number of remaining transfers which is $Z_i^P(t-1) - U_i(t-1)$, but we don't know how long each unit occupies a server. If the remaining work on each unit can be performed within the period, then $w_i(t) = Z_i(t) - Z_i^P(t-1)$. (This is obviously the case for instantaneous processes). On the other hand, the work which can be done on an "old" transfer unit during period t is bounded by $\frac{min(1,D_i)}{D_i}$ so that the amount of work which can be invested in all the "old" transfers is bounded by :

$$[Z_i^P(t-1) - U_i(t-1)] \cdot \frac{min(1,D_i)}{D_i}$$

Thus :

$$w_i(t) \geq z_i(t) - [Z_i^P(t-1) - U_i(t-1)] \cdot \frac{min(1,D_i)}{D_i}$$

By combining the two constraints and the requirement for non-negativity we get a conservative estimate for $w_i(t)$:

$$w_i(t) = \max \begin{cases} z_i(t) - [Z_i^P(t-1) - U_i(t-1)] \cdot \dfrac{min(1,D_i)}{D_i} \\ Z_i(t) - Z_i^P(t-1) \\ 0 \end{cases}$$

The complete model of the transfer relationships between activities which are characterized by finite, uninterrupted and fixed processing time and intra-period starting times is as follows :

1) Intensity :

$$z_j(t) \le \begin{cases} \dfrac{n_j}{D_j} \\ Z_i^P(t) - Z_j(t-1) \\ [Z_j^P(t) - U_j(t-1)] \cdot \dfrac{min(1,D_j)}{D_j} \end{cases}$$

2) Actual output :

Just as in our computation of w_i the cumulative information is not sufficient to determine the exact number of releasable transfers by time t. We can set the following bounds on $U_i(t)$:

$$Z_i^P(t - \lceil D_i \rceil) \le U_i(t) \le Z_i^P(t+1 - \lceil D_i \rceil)$$

Hereafter we shall use the following conservative estimate :

$$U_j(t) = Z_j^P(t - \lceil D_j \rceil)$$

3) Number of starts :

$$w_j(t) = \max \begin{cases} z_j(t) - [Z_j^P(t-1) - U_j(t-1)] \cdot \dfrac{min(1,D_j)}{D_j} \\ Z_j(t) - Z_j^P(t-1) \\ 0 \end{cases}$$

$$\hat{k}_j(t) = \left\lceil \frac{w_j(t) \cdot D_j}{min(1,D_j)} \right\rceil$$

$$Z_j^P(t) = \sum_{u=0}^{t} \hat{k}_j(u)$$

4.1) Upper bound on starts for complementary inputs :

$$Z_j^p(t) \le \min_i \lfloor V_{ij}(t-1)/\bar{a}_{ij} \rfloor \quad : \bar{a}_{ij} \ne 0$$

4.2) Upper bound on starts for additive inputs :

$$Z_j^p(t) \le \lfloor \sum_i V_{ij}(t-1)/\bar{a}_{ij} \rfloor \quad : \bar{a}_{ij} \ne 0$$

5) Capacity :

$$\sum_i a_{ki} z_i(t) \le x_k(t)$$

6) Sum of transfers :

$$\sum_j V_{ij}(t) \le U_i(t)$$

7) Non-negativity

2.5 The Relationships Between DLAAM and DLAAMP.

DLAAMP is an extension of DLAAM and evolved from the recognition that DLAAM as is, might be inaccurate if $D_i > 1$ and there are multiple servers. The derivation of DLAAMP gave us the opportunity to examine the implicit assumptions which underly DLAAM.

If we assume the coupling between a transfer unit and a server as assumed in DLAAMP, then the formulation of the production function of DLAAM might be inaccurate for non-instantaneous production processes. (If the activity requires some finite processing time we need an additional function to index the number of units which are available for transfer).

It is easy to show that DLAAMP with "intra-period" starts reduces to DLAAM in the limiting case of instantaneous production process :

$$\lim_{D_j \to 0} U_j(t) = Z_j^P(t)$$

Since : $U_j(t) \le Z_j(t) \le Z_j^P(t)$, we get that for instantaneous production processes $U_j(t) = Z_j(t) = Z_j^P(t)$.

As explained above, the amount of work invested in "new" starts at period t , denoted by $w_j(t)$, simplifies when $D_j \to 0$, to : $w_j(t) = Z_j(t) - Z_j^P(t-1)$ and the number of new starts, $\hat{k}_j(t)$, is the smallest integer larger or equal to this expression. By replacing $Z_j(t)$ by $Z_j^P(t)$, this constraint is becoming merely a definition of the incremental starts and can be removed.

Since : $U_j(t-1) = Z_j(t-1)$, for instantaneous processes, the intensity constraint can be written in the following DLAAM form :

$$z_j(t) \le \begin{cases} z_j^{\max} \\ Z_j^P(t) - Z_j(t-1) \end{cases}$$

and the output is the familiar expression :

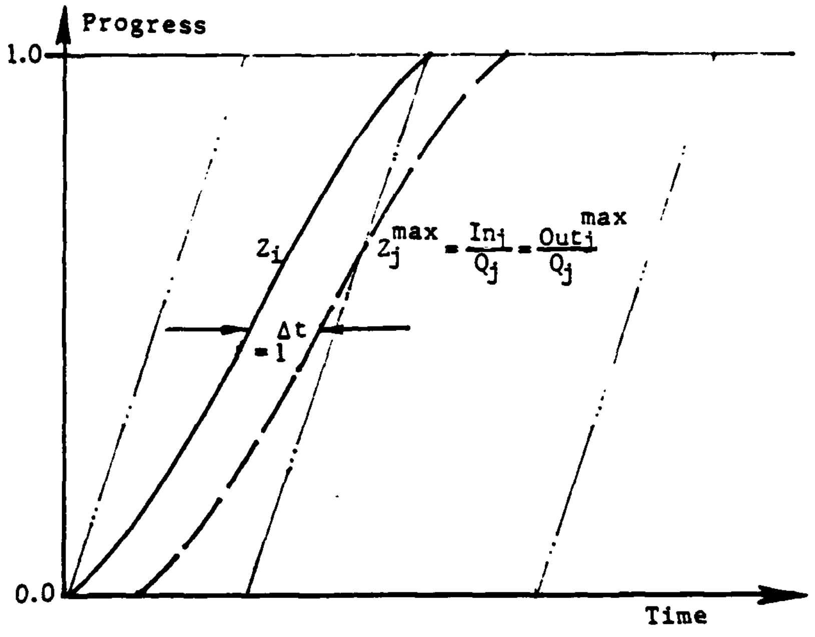
$$U_j(t) \leq Z_j(t)$$

Equations 4.1, 4.2, 5, 6, and 7 of DLAAMP, which are identical to the corresponding equations of DLAAM (see 2.3.1), remain unchanged.

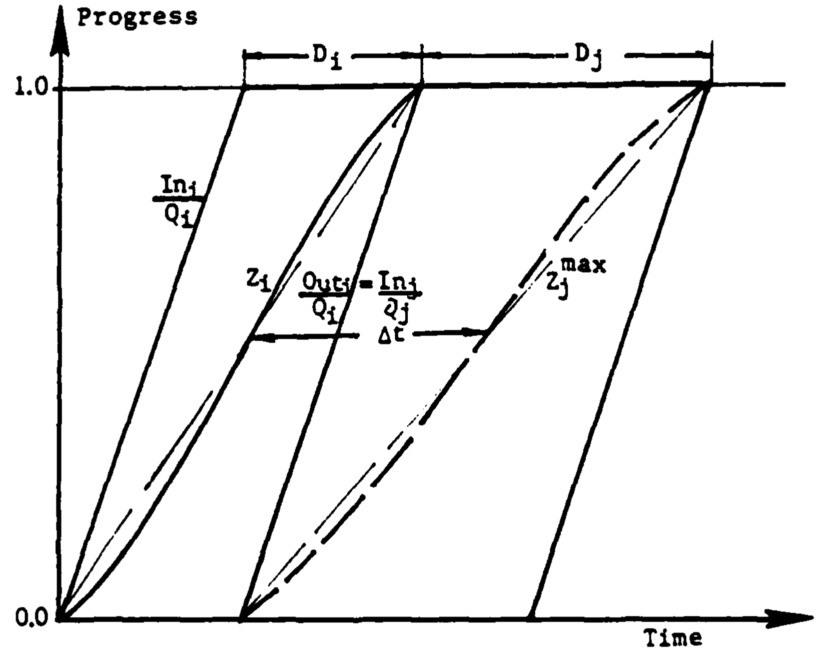
The assumption of coupling is appropriate in environments where changing the server is impossible or requires a significant set-up. On the other hand, some environments allow (unrestricted) change of the servers and it is easy to show that if $D_i \leq 1$, *the units can be sequenced* such that the output in each period equals $\lfloor Z_i(t) \rfloor$. Under this discipline DLAAM is accurate for the whole range $0 \leq D_i \leq 1$.

It is also interesting to note that the assumption of coupling between a server and a transfer-unit is relevant only in a multi-server environment. When an activity consists of a single server the DLAAMP model reduces to DLAAM where $z_i^{\max}(t) \leq \frac{1}{D_i}$.

The merits of DLAAMP are both theoretical and practical. Through this model we show that in general, dynamic modeling of production systems requires separate indexing functions for the application of service type resources and the application of intermediate products. An interesting observation is that Dynamic Models introduced in the classic text-books of production such as Johnson & Montgomery [1974], Silver & Peterson [1985], Hax & Candea [1984] and others assume implicitly instantaneous production processes. The practical contribution is in identifying and extending the range for which we can develop reasonably accurate activity analysis models. In figure 2.5 we compare the behavior of a system consisting of two ordered activities under the existing model (DLAAM) and the proposed one (DLAAMP).



a) modeled by DLAAM



b) modeled by DLAAMP

Figure 2.5: Transfer relationships of a production system with finite processing-time.

3. Aggregation Approaches .

3.1 General approach :

The purpose of the aggregation is to provide aggregate information to support managerial decisions. A good aggregate model should provide management with information which is relevant to its level of interest, with sufficient accuracy and require much less effort than the detailed model. In this chapter we shall introduce several approaches for aggregation of production networks. Though these approaches vary in their scheme of aggregation as well as in their applications, the process of aggregation has the same general structure as illustrated at figure 3.1 : The basic stages involved are the following :

a. An "aggregation scheme" is applied to the original network, converting it into a smaller aggregate network.

b. The *aggregate network* is simulated or optimized using some model of production. Most of the models used by the approaches that we shall introduce, are models which were developed for *detailed networks* and view the aggregate network as if it were a detailed network. Most of the models assume the same structure and properties for the detailed and the aggregate networks. The models which belong to the class of parallel aggregation (Leachman & Boysen, and ours) are aggregating detailed activities of the "project" type, into aggregates which are treated as activities of the "processing" type.

c. Aggregate results: The result of the previous two stages is an "answer" to the problem which was investigated . An inherent result of the process of aggregation is the "loss of identity" of detailed activities which are grouped together. This loss might be tolerable for managerial decisions of the tactical level where management is interested in aggregate measures, but not for operational decisions. The user must be very careful in examining to what extent this "loss of identity" effects the quality of the results for specific decisions.

We introduce several schemes of aggregations, each of which has its typical applications. We shall emphasize the properties of each of these aggregation schemes and the kind of decisions supported by it.

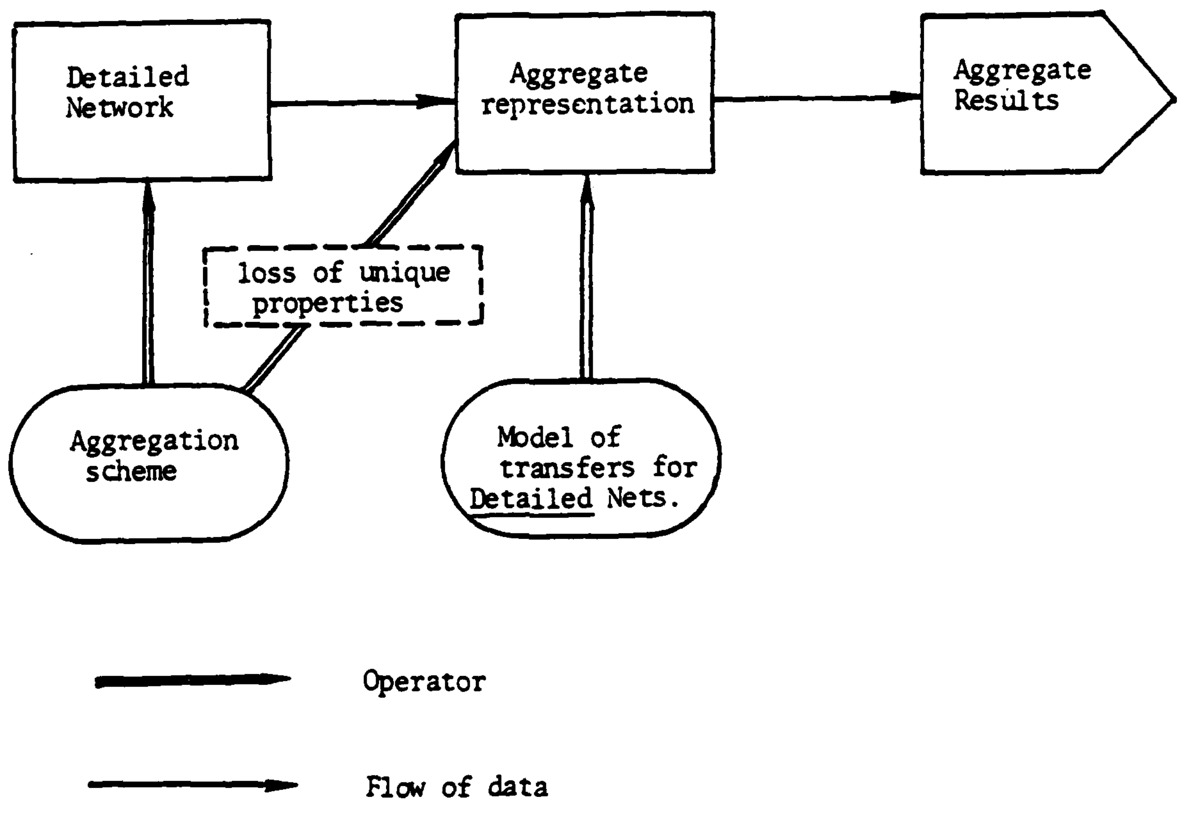


fig. 3.1 : The aggregation of detailed networks .

3.2 Aggregation by product :

The most obvious aggregation scheme is the "aggregation by product". In this scheme we attempt to reduce the size of the network by representing several batches of *similar* products, as a single aggregate batch of the "generalized" product. We can learn a great deal about the requirements of "similarity" by investigating the aggregation of a *general linear program of production.*

Suppose, we want to find the mix of products which gives the highest revenue under a given set of capacities. Let x denote the vector of quantities which are produced *and* sold of the products 1 to n, and r denotes the vector of revenues from each unit of product.
$\{a_{kj}\}$ is the matrix of the technological coefficients of production, where a_{kj} is the amount of resource k required for the production of a unit of the j'th product. (The matrix will be denoted by A) The capacity of the resources will be denoted by the vector b .

The system is modeled by the following linear program :

$$MAX : r^T \cdot x$$

Subject to :

$$A \cdot x \leq b$$

$x_j \geq d_j$, where d_j denotes the demand of product j

$x_j \geq 0$ for each j

Now, let's aggregate columns k and l by an aggregate m such that each unit of the aggregate is "equivalent" to α_j units of product j where $j \in (k,l)$. Each aggregate unit is assigned aggregate technological coefficients and revenue per unit.

The aggregate presentation of the system will be :

$$MAX : \bar{r}^T \cdot z$$

subject to :

$$\bar{A} \cdot z \leq b$$

$$z_m \geq \frac{d_k}{\alpha_k} + \frac{d_l}{\alpha_l}$$

$z_j \geq 0$ for each j

where $\bar{r}$, z and $\bar{A}$ are the aggregate revenues, products and technological coefficients accordingly .

The conditions for an *optimal aggregate solution* are :

1) Primal feasibility
2) Dual feasibility (y is the vector of "shadow prices")
3) Complementary slackness :

$$z^T[\bar{A}^T \cdot y - \bar{r}] = 0$$

$$y^T[\bar{A} \cdot z - b] = 0$$

The conditions for an *optimal disaggregated solution*, given the optimal aggregate solution are as follows :

choose $x_k, x_l \geq 0$ such that the disaggregate revenue is maximized :

$$MAX : r_k x_k + r_l x_l$$

Subject to :

$a_{ik} x_{ik} + a_{il} x_l \leq \bar{a}_{im} z_m$, for each i .

$x_j \geq d_j \ , j = (k, l)$

Algebraic analysis of the requirements for disaggregation yields the following observations :

For any pair of tight constraints, the following ratio must hold :

$$\frac{a_{tm}}{a_{sm}} = \frac{a_{tk} x_k + a_{tl} x_l}{a_{sk} x_k + a_{sl} x_l}$$

It is easy to show that this relation is simultaneously maintained in all the pairs of constraints for which $a_{ik} = \gamma a_{il}$ and $\bar{a}_{am} = \delta a_{il}$, where γ and δ are constants.

b) The proportional mix is required only for resources which are tight . As a conclusion, we can exclude from the model resources which are certainly non-bounding without risking our ability of disaggregation.

For a rigorous discussion of the bounds on errors, due to aggregation of columns and rows see Zipkin [1980], Mendelssohn [1980].

Another practical requirement, resulting from the batching procedure is that detailed batches which are grouped together should have close due dates. (otherwise we accumulate products of different production periods and cause non-necessary production peaks).

The reader should be aware that the previous discussion did not take into account sequencing constraints. Thus, it is most appropriate for a single echelon production system.

This aggregation model is very useful in several industries, but as we shall show later, the aggregation by product can be extended to a multi-echelon aggregation using the approach of "parallel aggregation".

3.3 Aggregation by modules .

One of the most widely used aggregation techniques is aggregation by modules . In this approach, a subnetwork, usually related to a certain assembly or major operation, is collapsed into a single representing activity. (For a comprehensive discussion see Kasper [1983]).

The main advantage of this approach is that grouping detailed tasks into a meaningful aggregate has a natural organizational appeal. However, in order to represent the subnetwork by the single aggregate we must assume some underlying schedule of the inner activities. This predetermined schedule imposes two major drawbacks for dynamic modeling :

a. The underlying schedule is not necessarily optimal or even feasible under given capacities of scarce resources.

b. The mix of resources that are consumed by the aggregate is varying with time according to the requirements of the internal activities, thus the production function of the system can not be expressed in terms of a convenient index for the rate of the utilization of resources.

This type of aggregation preserves the precedence relationships of the network. Under the assumption of sufficient resources (which should be verified), this approach provides a convenient way for estimating lead-times and due-dates.

3.4 Parallel aggregation .

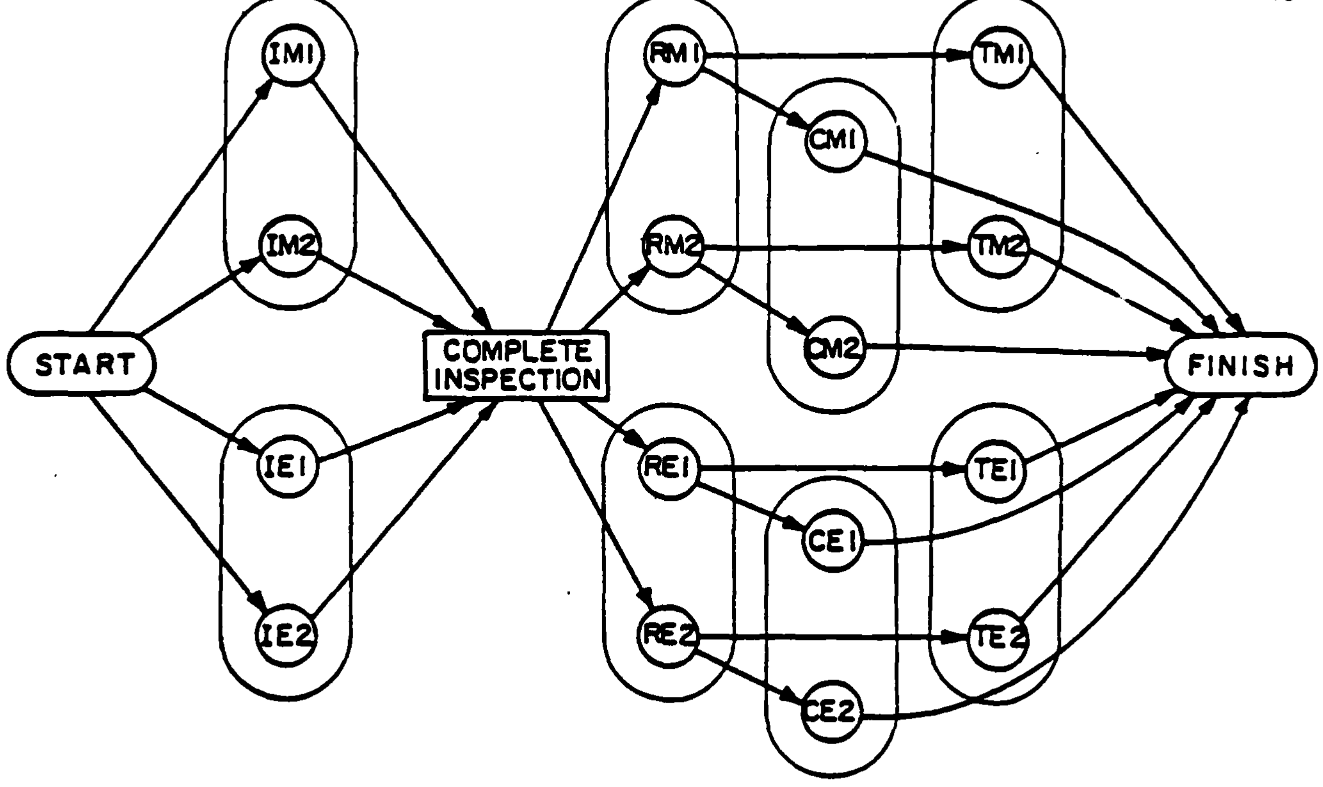
The idea of parallel aggregation was first introduced by Leachman & Boysen [1982]. In this approach activities which use a similar mix of resources and conform to certain structural restrictions are represented by an aggregate activity for the purpose of aggregate planning. The investigation of the required properties for aggregation of several products (3.2) can be extended to aggregation of activities, for which the same conclusions hold :

a. Activities which are grouped together use the same mix of resources. (proportionality of technological coefficients).

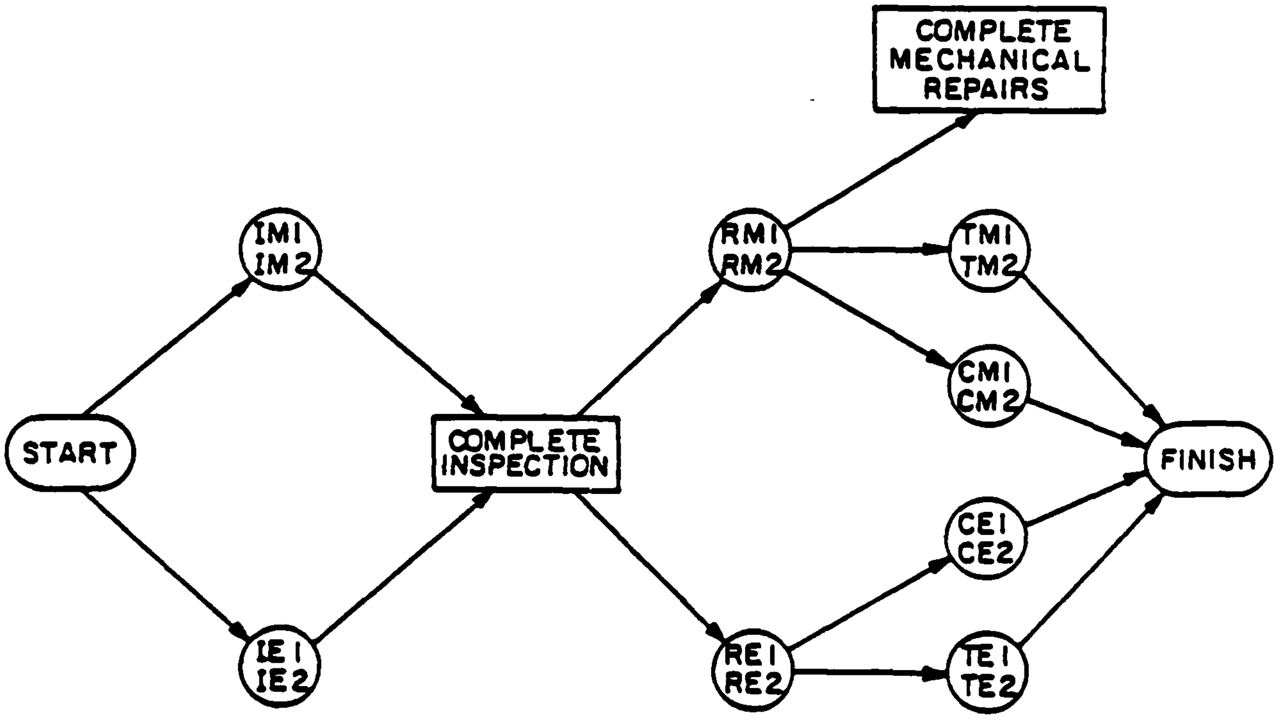
b. The requirement of similar mix applies only to scarce resources. We assume that other resources will not cause a problem at the stage of disaggregation.

This scheme of aggregation enables us to model the *aggregate network* by using *proportional or semi-proportional production functions*. The models which utilize this approach attempt to preserve the precedence structure of the network and provide good estimates for the rates of the usage of resources.

An example for parallel aggregation is illustrated in figure 3.4 , where we assume that the mix of resource required for each type of activity is identical. (this example was taken from Leachman & Boysen [1982]).



EXAMPLE - DETAILED NETWORK



EXAMPLE - AGGREGATE ACTIVITY NETWORK

Figure 3.4: An example for Parallel Aggregation

4. Parallel Structural Reduction.

4.1 Structural Requirements .

In chapter 3.1 we introduced the general process of obtaining information for managerial decisions, by replacing the detailed network, with an *aggregate* one. We also introduced several approaches for aggregation. In this chapter we shall focus on the "structural reduction" of parallel aggregation.

The requirements for the structural reduction are derived from the properties of the models which we intend to apply to the aggregate network. The *aggregate* network must require the following characteristics :

a. Directed and acyclic network .

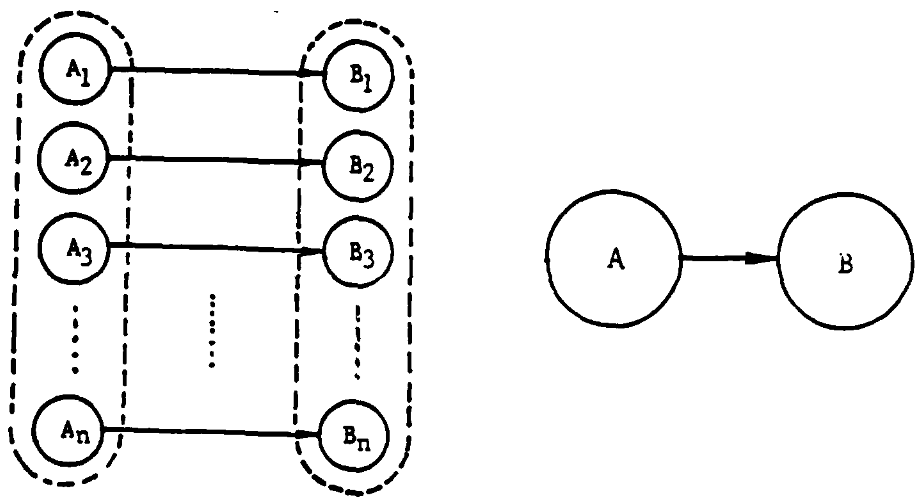
b. Activities will have proportional or semi-proportional production functions. This requirement implies that the underlying detailed activities which are grouped together, will have the same mix of resources. (By mix we mean both types of resources, and proportions of technological coefficients.)
For practical purposes we might relax the similarity of the mix for non-scarce resources. (See 3.2)

The basic "structural relationships" were first introduced by Leachman & Boysen. We extend their model which was designed for Project Networks to include also Processing Networks and introduce the need for distinction between Project Networks and Processing Networks.

4.2 Structural Relationships :

4.2.1 One to One :

Let activities of type *A* (a certain mix of resources) transfer their intermediate products only to activities of type *B* (another mix of resources), as demonstrated in figure 4.2.1-a . The aggregate representation of this grouping is illustrated in figure 4.2.1-b . The aggregation scheme of this type of relationship is suitable for both processing and project networks.



4.2.1. a - Detailed transfer relationship

4.2.1. b - Aggregate representation

fig 4.2.1 : Detailed and Aggregate "one to one" relationship.

It is clear from this illustration that activities A1,A2 and A3, as represented by their aggregate *A*, have lost their unique properties . The more "identical" these activities are (similar total requirements of resources and similar rates of operation) they can be better represented by their aggregate *A* .

4.2.2 Multiple Successors :

Let activities of type A transfer their intermediate products to both activities of types B and C . We shall distinguish between two cases :

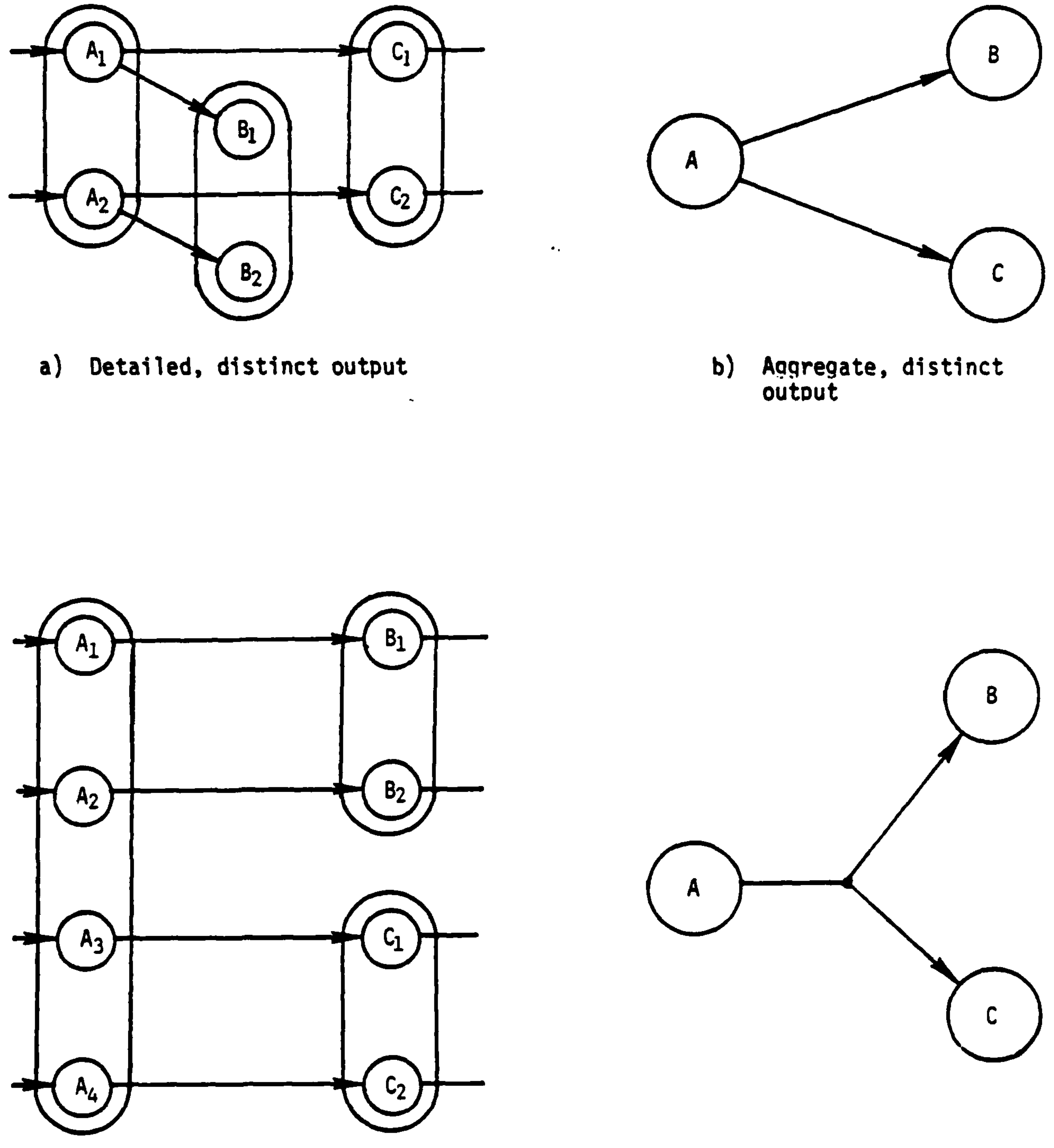
a. *Distinct outputs:* The aggregate A consists of activities which release outputs simultaneously to activities of types B and C . A typical situation is when activities of type A in a "Project Network" precede activities of both types B and C . The detailed sub network is illustrated in figure 4.2.2-a while the corresponding aggregate sub-network is illustrated in figure 4.2.2-b . This structure of outputs is typical to Project-Networks.

Modeling Issues :

Since we assume distinct products which are produced by A simultaneously and transferred in parallel to activities B and C, the "mass conservation" assumed by the transfer relationships of our models (regardless which one) should not "double count" the transfers to B and C. (See figure 4.2.2-a)

b. *Shared outputs :* The aggregate A consists of activities which transfer their outputs to activities of type B, type C , or to both. The detailed and aggregate presentation of this case are illustrated in figures 4.2.2 c and d respectively. When the assumption of "shared outputs" is appropriate, the output of aggregate A equals its inventory plus the the sum of the transfers to B and C. This is the structure of output assumed in Processing-Networks.

So far, we have used examples with a single predecessor and two followers. It is obvious that the definitions hold for as many successors as applicable for the particular network .



a) Detailed, distinct output

b) Aggregate, distinct output

c) Detailed, shared output

d) Aggregate, shared output

Figure 4.2.2: Detailed and Aggregate multi-successor structure.

4.2.3 Multiple Predecessors :

The relationship of multiple predecessors is very similar to the previous one. Here we have to distinguish between **distinct** inputs (figure 4.2.3 a and b) and **pooled** inputs . The detailed network and the corresponding pooled input aggregate are illustrated in figure 4.2.3 c and d .

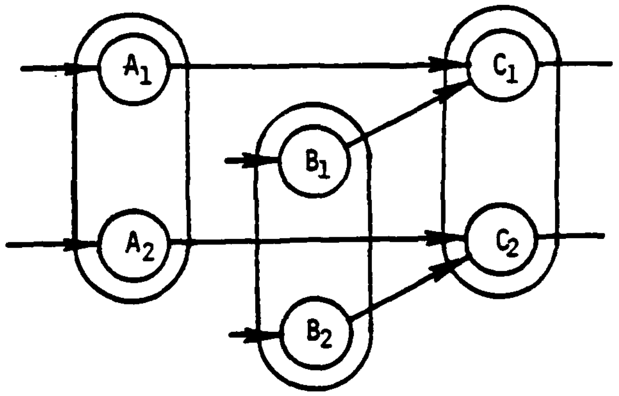
The relative weight of sub-aggregates :

Let's consider the common case where aggregate A supports only a part of B, denoted by B_A. We shall define the weight of the relative part of B supported by A as W_{BA}, where :

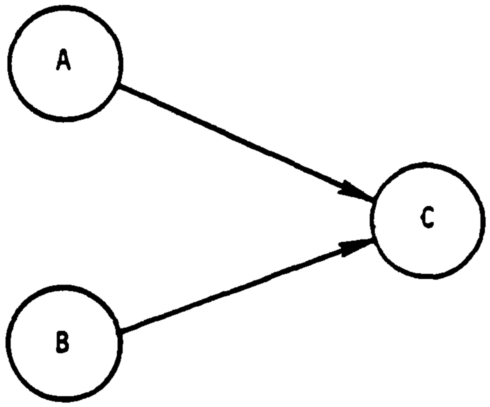
$$W_{BA} = \frac{\sum_i a_{ki} \mid i \in B_A}{\sum_i a_{ki} \mid i \in B}$$

In the transformation from detailed into aggregate transfer relationships, we shall use the approximation that from each input unit which arrives to aggregate B, W_{BA} of the input supports B_A.

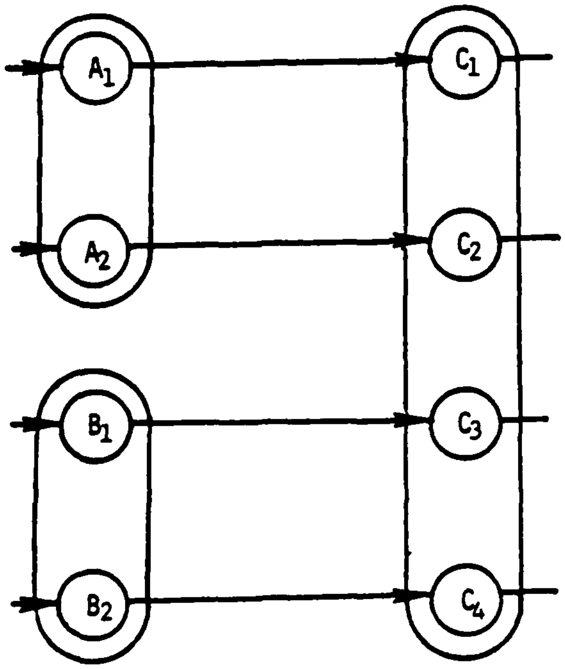
Accurate modeling of *Project Networks* requires further partition of the aggregates. Let an inbound sub-aggregate be the group of detailed activities within the aggregate, which posess the same set of *aggregate predecessors*, and the outbound sub-aggregate the group which posess the same set of *aggregate followers*. The level at which any "inbound" sub-aggregate can operate is determined by the predecessor which provides the lowest "level of support". In order to avoid double counting of distinct outputs, it is also necessary to identify the "outbound" sub-aggregates. These additional details decrease the attractiveness of the aggregation. In order to overcome this deficiency, we shall propose an **approximation** which uses the sub-aggregates merely for the computation of "compensation" factors. These factors enable us to approximate a Project Network by an "equivalent" *corrected* Processing Network, where the relationships are at the aggregate level. This approximation is described in sections 5.2.2 and appendix A .



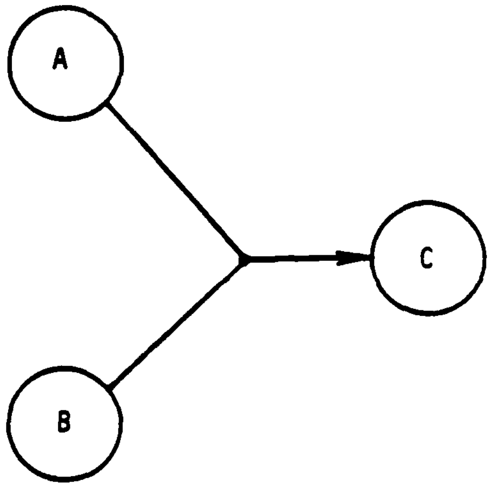
a) Detailed, distinct (complementary) inputs



b) Aggregate, distinct (complementary) inputs



c) Detailed, pooled (additive) inputs



d) Aggregate, pooled (additive) inputs

Figure 4.2.3: Detailed and Aggregate multi-predecessor structure.

5. Models of Dynamic Aggregate Networks.

5.1 Modeling transfers by the "relative progress".

The model that we shall introduce in this section is the first one to propose "parallel aggregate networks" and solve dynamic aggregate problems by a Linear Program formulation. The model is due to Leachman & Boysen [1982]. (See also Boysen [1982]).

The "Leachman-Boysen" model was developed for modeling of the overhaul of ship in a shipyard. The underlying detailed network is of "project" type activities which are called "Key-Ops". The aggregation scheme of the Key-Ops is as presented in the previous chapter, sections 4.1-4.4 . The model is applied to the aggregate representation of the overhaul-network.

The model is a modification of the DLAAM, motivated by the previously discussed observation that the cumulative intensity does not represent correctly the output of an activity. Leachman & Boysen introduced an experimental function, the *relative progress*, to represent the transfer relationships between the activities. In the next paragraphs we shall show the derivation and rationale of this function. We shall use the following concepts :

Mode of Operation : The "history" of the work performed by an activity, represented by the cumulative intensity $Z_i(t)$.

Window of Operation : Each activity can operate only within a bounded range of "modes" . The fastest mode of operation is achieved when the whole network operates in the early mode (Early Start and highest intensity) without capacity constraints. This mode is denoted by $Z_i^E(t)$.

The "late mode" where each activity operates as late as possible **without affecting the due dates of the network** will be denoted by $Z_i^L(t)$.

Any activity is bound to operate between these two curves.

$Z_i^L(t) \leq Z_i(t) \leq Z_i^E(t)$. The "area" between the two bounding curves is referred to as the "window of operation".

It is easy to show (Boysen [1982] , Hackman [1983]) that the intensity of an aggregate is the weighted sum of the intensities of the included detailed activities, which enables us to obtain the bounding modes of operation for the aggregates as well. The "window of operation" is illustrated in figure 5.1-1 .

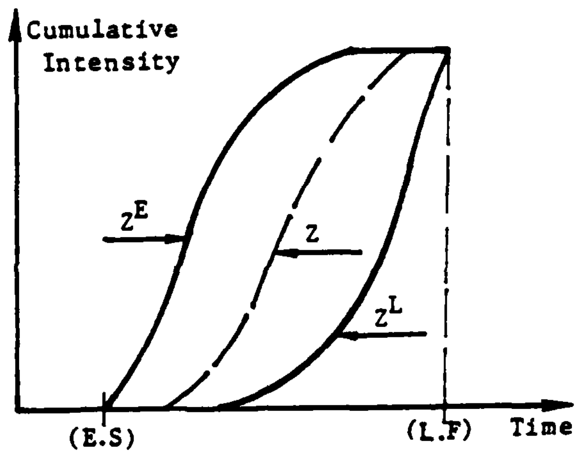


fig 5.1-1 : Window of Operation

Relative Progress : Leachman & Boysen defined a new function which attempts to represent a relationship between the output of the predecessor and the possible outputs of its followers. The relative progress is defined as follows :

$$p_i(t) = \begin{cases} 0 & t < ES_i \\ \dfrac{Z_i(t)-Z_i^L(t)}{Z_i^E(t)-Z_i^L(t)} & ES_i \le t \le LF_i \\ 1 & t > LF_i \end{cases}$$

Slack of Operation : Leachman & Boysen defined the "slack of operation" as the area between the curves of the *normalized* Early and Late modes. This definition is a generalization of the usual definition of slack in CPM networks (the difference between Late Start and Early Start). Note that the new definition relates two curves while the traditional one relates two points of time. A *partial slack* will be defined as :

$$\sigma_i(t) = \frac{\int_{ES_i}^{t} [Z_i^E(u) - Z_i^L(u)] \cdot du}{\int_{ES_i}^{LF_i} [Z_i^E(u) - Z_i^L(u)] \cdot du}$$

Leachman & Boysen proposed the following *relative* transfer relationship between an activity i and its follower j :

$$p_i(ps_i) \geq \bar{a}_{ij} p_j(ps_j)$$

where ps_i and ps_j are two points of time which correspond to *equal partial slack* for the predecessor and the follower accordingly . This transfer relationship is intuitively justified if we note that in both extremes, when the predecessor operates in either Early mode or in its Late mode, the follower's *earliest* mode is the same as the mode of the predecessor . The "space" between these extremes is assumed to be interpolatable . For more detailed justification see Hackman [1983] .

Computational Simplification :

A usual dynamic formulation such as the models which were introduced in chapter 2 requires setting the values of the control variables at each point on the time grid . This kind of formulation results in a very large number of variables and constraints and might cause many practical "problems" to fall outside of present capabilities of today's computers . The following observations enabled the authors to reduce the number of "samplings" and thus the size of the formulations :

a. Each activity (or aggregate) has an *predetermined window of operation*. The same holds for the *partial slack*. This property leads to the next observation :

b. Each activity (or aggregate) has a certain relevant time frame, between the early-start and the late-finish, in which it can operate . The values of the variables which correspond to a certain activity are fixed during the periods which are external to the window of operation . This property enables us to reduce the number of variables and the corresponding constraints by a significant amount .

c. The operation of practical activities can be usually divided into three phases :

1. Phasing in (increasing rate)
2. "Production" (constant rate)
3. Phasing out (decreasing rate)

Modeling :

The model assumes the following additional simplifications :

a. The relative progress during the first and the third phase are constant (not necessarily equal)

b. The intensity during the second phase is constant .

Since the condition of relative progress is maintained at points of time which correspond to equal partial slack, the authors divided each activity into three intervals where the partition is done at fixed points of partial slack . The balance of the relative slack is maintained at the finishing points of each phase .The resulting model is as follows :

Let m denote phase number and F_i^m the time which corresponds to the finish of phase m . We shall also use $[m]$ for phase m.

intensity bound :

$$0 \le z_i[2] \le z_i^{\max} \quad \text{for each i}$$

relative progress bound :

$$0 \le p_i[m] \le 1 \quad m=1,2,3 \quad \text{,for each i}$$

relative progress balance :

$$\sum_i \bar{a}_{ij} \cdot p_j(F_j^{m}) \le \sum_j W_{ji} \cdot p_i(F_i^{m}) \quad m=1,2,3$$

intensity continuity :

$$p_i[1] \cdot \sigma_i(F_i^{1}) + (F_i^{2} - F_i^{1}) \cdot z_i[2] - p_i[3] \cdot \sigma_i(F_i^{2}) = Z_i^{L}(F_i^{2}) - Z_i^{L}(F_i^{1})$$

capacity :

$$\sum_i a_{ik} z_i(t) \le X_k(t) \quad t \in \{F_i^{m}\}$$

Advantages of the model :

The Leachman-Boysen model has a very significant "pioneering" contribution to parallel aggregation and to the modeling of Dynamic Aggregate Networks. Considering the results of the validation tests, it also seems to be practical as a design tool for the environment of the shipyard. Its specific contributions are as follows:

1. First one to propose and model parallel aggregation .

2. First to recognize the need for distinction between the amount of work which was invested in an activity (or aggregate) and the output of the activity.

3. First to propose the "three phases" scheme for the reduction of the size of the Linear Program formulation . (A similar idea was introduced as a basic concept of "discrete events simulation" where "dead" time-periods are skipped .)

Limitations of the model :

The main limitation of the model is that it was "tailored" to the specific characteristics of the shipyard . Though the tests done by the authors seem to give satisfactory results , we might expect the following shortcomings in a more general framework :

1. The transfer relationship is not based explicitly on the output of the activities or the aggregates . Though true in both extremes of Early and Late modes, it was not proved that the relative progress relationship always can be accurately interpolated in between. (The Early and Late modes correspond to certain schedules in both (or the multiple) aggregates. Intermediate modes might be obtained from various internal schedules in each aggregate . Furthermore, the Early and Late modes are obtained from the analysis of uncapacitated networks and might be unattainable in practical, resource-constrained situations.)

2. The model of the "three phases" is a very good approximation for *detailed* activities. When we model aggregate activities, each of which represents many detailed activities, the "shape" of $Z_i(t)$ is no longer predictable. However, one can expect "trapezoidal" behavior *if* any of the following circumstances occur :

a) A resource which is occupied mainly by a certain aggregate operates at its upper capacity.

b) Each aggregate has various sources of inputs (from its predecessors) such that we can assume a *large flow of independent inputs*. In this case one can expect that the aggregate operates, in average, at a constant rate, except for some "phasing" in and out periods.

3. A practical application of the model requires the preliminary effort to generate the two extreme modes of operation, the Early and the Late modes.

5.2 Aggregate Model Based on DLAAMP .

5.2.1 Conceptual Introduction

The DLAAMP model for detailed networks of the "processing" type was introduced in section 2.4 . In this section we shall introduce the application of this model for *aggregate* networks of the "processing" type .

Conceptual foundation :

The application of the DLAAMP to aggregate networks is based on the *parallel aggregation* scheme. Let the detailed network consist of activities of the "project" type (fits Project Networks and certain Processing Networks. The relaxation of this assumption will be discussed later). If we aggregate activities which are similar in mix, duration and resource requirements we can regard the included activities as transfers produced by the aggregate. Under the assumption of similarity all we need to know is that a detailed activity of type A, when completed, releases an activity of type B . This behavior is analogous to the model of the detailed DLAAMP where a transfer completed by activity i can be processed by activity j . Figure 5.2 illustrates the concept of "aggregate DLAAMP" . (We consider the event of completion of a detailed activity as equivalent to the physical transfers, discussed in chapter 2 .)

The assumptions of the model as stated so far seem to be very restricting . However , we can propose several situations where the model applies without further relaxations :

a. *Processing environment* : Suppose we deal with a situation where we produce multiple batches of the same products and same batch size, but with different due-dates. The detailed network consists of completely parallel and identical "strings" of activities. This network can be represented by a single string of aggregates where the due-dates affect the requirements for the "final output" of the *aggregate* string.

b. *Construction environment* : Suppose similar construction projects are done in several locations and share the same resources. Here again it is easy to see that the parallel detailed network can be reduced to a single string of activities.

As demonstrated so far, an underlying assumption in applying DLAAMP to model the transfer relationships between aggregates is the similarity (theoretically, identity) between activities of the same type. In order to expand the ability of our model to deal with practical problems, we shall have to relax this assumption of "complete" similarity, to similarity in mix of resources and to define "average" activities. Let aggregate s consist of n_s detailed activities (denoted by i). Then the characteristics of the "average" activity are as follows :

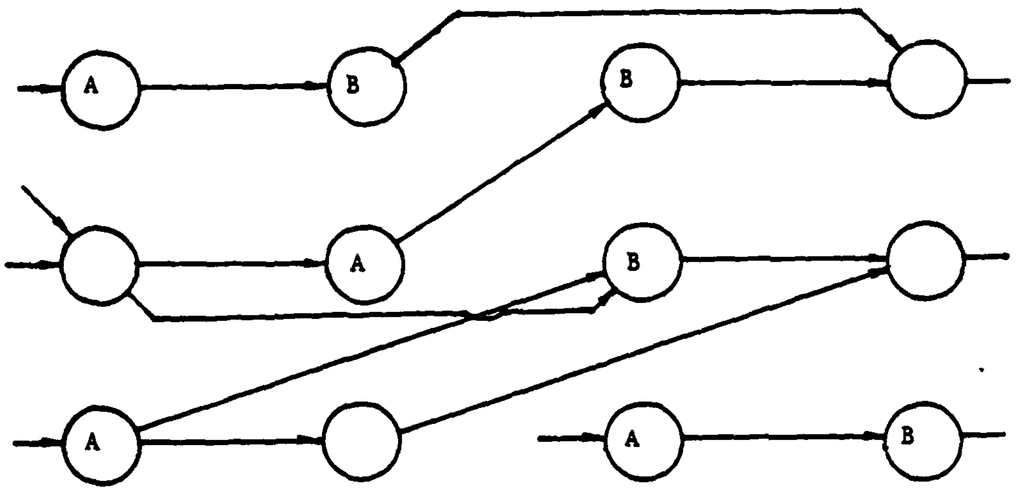
k'th resource requirement per (average) detailed activity :

$$\hat{a}_{ks} = \frac{\sum_{i \in s} a_{ki}}{n_s}$$

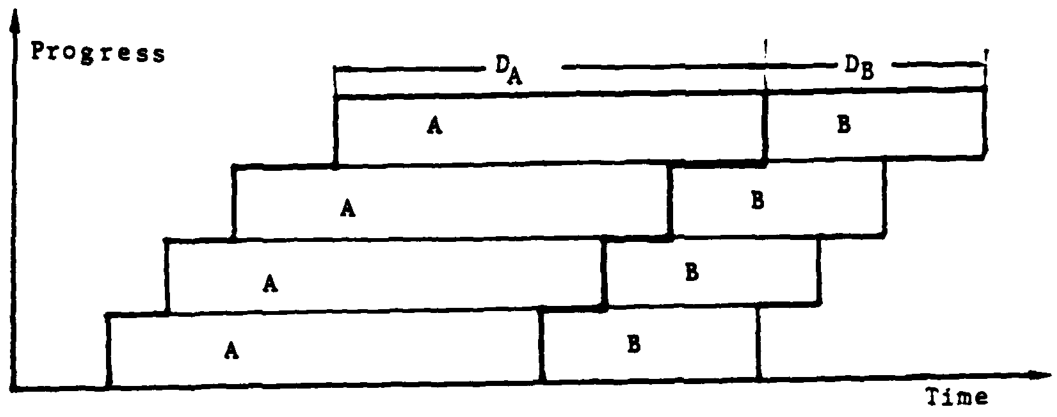
"average" duration :

$$\hat{d}_s = \frac{\sum_{i \in s} a_{ki} d_i}{\sum_{i \in s} a_{ki}}$$

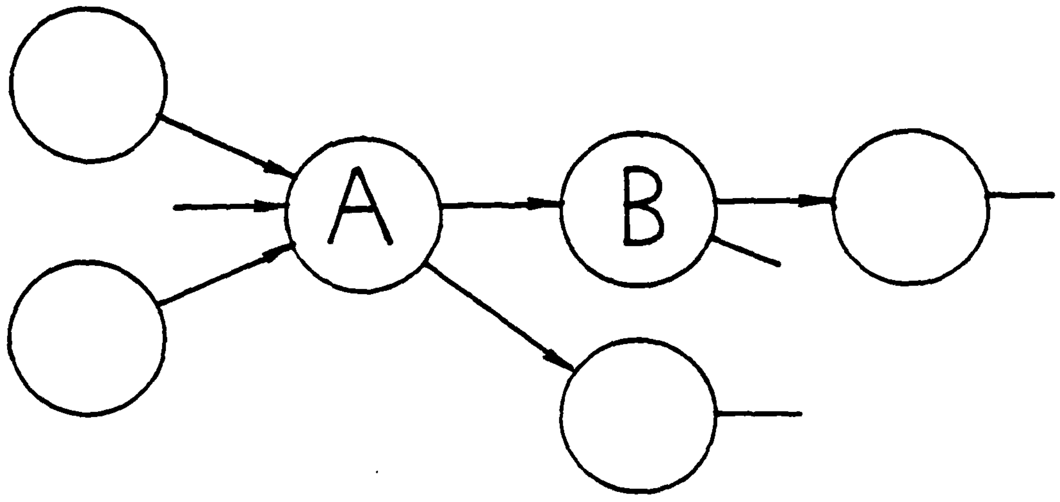
Under the assumption of similarity in mix, we can arbitrarily choose the resource which we use for the computation of the average duration, as long as this resource is used by the aggregate.



a) Detailed Network



b) Transfer Relationships



c) Aggregate Representation

Figure 5.2: The transformation from detailed network to aggregate representation.

5.2.2 "DLAAMP" Aggregate Project Networks Model

The model of DLAAMP as presented in section 2.4 dealt with the transfer relationships between activities of the "processing" type. In this section we shall assume that the aggregates are of the "processing" type and the detailed activities have *distinct* (complementary) inputs and outputs. We shall regard the completion of a detailed activity as equivalent to the completion of a transfer unit in the detailed model.

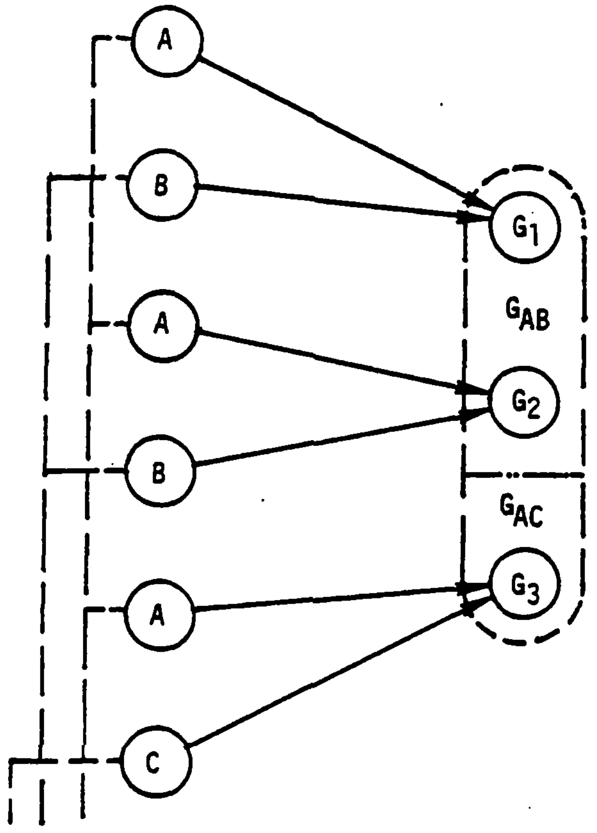
Definitions :

$I(j)$	The set of aggregates which precede aggregate j .
$J(i)$	The set of aggregates which follow aggregate i .
n_j	The number of detailed activities included in aggregate j .
n_{ij}	The number of detailed activities included in aggregate i , which deliver output to aggregate j .
n_{ji}	The number of detailed activities included in aggregate j , which receive input from aggregate i .
V_{ij}	The number of "average" detailed activities completed by aggregate i and allocated to be followed by activities which are included in aggregate j . (Analogous to "physical" transfers in the model of detailed networks.)
NS_i	Number of aggregate servers available at activity i .

The situation that we attempt to model is quite complex and should be examined very carefully :

a. The aggregate network consists of *aggregates* with *shared* inputs and outputs.

b. The sharing *sub-aggregates* (each one with a defined set of predecessors) have *distinct* inputs and outputs when we aggregate *Project Networks* . (We shall discuss Networks in a later section). The structure of *distinct* inputs and outputs implies that the level of operation of each sub-aggregate is determined by the *minimal level of "support"* that it gets from its predecessors. (see fig. 5.2.2)



$$U_{G_{AB}}(t) \leq min[V_{AG_{AB}}(t+1 - D_G), V_{BG_{AB}}(t+1 - D_G)]$$

$$U_{G_{AC}}(t) \leq min[V_{AG_{AC}}(t+1 - D_G), V_{CG_{AC}}(t+1 - D_G)]$$

fig. 5.2.2 : "Level of support" of sub-aggregates.

c. The level of operation of the whole aggregate is determined by the weighted sum of the levels of operation of its sub-aggregates.

d. In order to simplify "book-keeping", we shall assume that parallel outputs of detailed

activity l are *assigned* to all the detailed followers (m,...,m'), such that $V_{lm}(t)=,...,=V_{lm'}(t)$.

e. In order to obviate multiple-counting of the parallel (distinct) output of each detailed activity, we shall define a "compensation factor" which ensures, at least, that the sum of the *final* outputs of aggregate i, will equal n_i. The outputs sent to the follow-on aggregates are weighted by these factors. We propose and prove the appropriateness of such coefficients in appendix A . The coefficient which multiplies the output delivered from aggregate i to aggregate j will be denoted by β_{ij}.

f. All the assumptions which were introduced in chapter 2 section 4 apply here, thus the formulation is essentially similar. Though the term of aggregate servers is quite ambiguous when we model transfer of activities, the same greedy assumptions hold. In this model a "server" will represent the *group of servers, operating simultaneously, to finish the task of a detailed activity $l : l \in i$, in D_i time units.* For the sake of simplicity we shall assume that D_i is an integer number of periods.

Let Λ_{is} denote the set of aggregate predecessors which support sub-aggregate s of aggregate i .

As explained above, the output of an aggregate is approximated by the weighted sum of the "deliveries" :

$$U_i(t) = \sum_{m \in J(i)} \beta_{im} \cdot V_{im}(t) + I_i(t)$$

The contribution to the whole aggregate, from an input of sub-aggregate s is as follows :

Each unit of input which arrives from aggregate i to aggregate j should be multiplied by the appropriate "explosion factor" $\frac{n_j}{n_{ij}}$, but since it supports only sub-aggregate s , it should be scaled by the *relative weight* of sub-aggregate s of aggregate j , denoted by W_{j_s}. (defined in chapter 4)

Model formulation :

The intensity of aggregate j is determined by :

$$z_{j_s}(t) \le \max \begin{cases} \dfrac{NS_{j_s}}{D_j} \\ Z_{j_s}{}^{P}(t) - Z_{j_s}(t-1) \end{cases}$$

$$z_j(t) = \sum_s z_{j_s}(t)$$

The output of aggregate j :

$$U_j(t) = Z_j^P(t+1 - D_j)$$

Where the number of starts is determined by :

$$Z_{j_s}{}^{P}(t) \le \min_{l \in \Delta_u} [V_{lj_s}(t-1) \cdot \frac{n_j}{n_{ij}} \cdot W_{j_s}\ ]$$

$$Z_{j_s}{}^{P}(t) = \min \begin{cases} \lceil z_{j_s} \cdot D_j \rceil + U_{j_s}(t-1) \\ Z_{j_s}{}^{P}(t-1) \end{cases}$$

$$Z_j^P(t) = \sum_s Z_{j_s}{}^{P}(t)$$

Allocation of the output :

$$U_j(t) = \sum_{m \in J(j)} \beta_{jm} V_{jm}(t) + I_{j(t)}$$

Output monotony :

$V_{ij}(t) \ge V_{ij}(t-1)$,for each ij .

Capacity constraints :

$$\sum_i a_{ik} \cdot z_i(t) \leq X_k(t) \quad k=1,\dots,K$$

Even though approximate, this set of constraints is too complicated for practical purposes. In the next sections we shall propose a further approximation which will enable us to work at the level of the aggregates. This further approximation will be based on the following model for Processing Networks.

5.2.3 Transfer relationships for Processing Networks.

The distinction that we made between Processing Networks and Project Networks was mainly in the nature of the inputs and outputs at both systems. Our assumption was that *Processing Networks have shared inputs and outputs.* This assumption simplifies a great deal the transfer relationships between the aggregates .

Since *both* levels , the aggregates and the sub-aggregates have *shared* inputs and outputs , there is no reason in aggregation of Processing Networks to refer to the level of sub-aggregates. Transfer relationships are strictly between the aggregates.

Relaxation of the integral domain constraints :

As stated before, accurate modeling requires that activities of the "project" type will be represented as *discrete* outputs of the relevant aggregate . The other extreme will be when the detailed activities are of the "continuous flow" type which obviously implies that the aggregates will also be of the "continuous flow" type . An intermediate case will be when the *detailed* activities have "discrete flow" transfers . However, for practical reasons, we recommend that for Processing Networks the integral domain constraints will be relaxed if it simplifies the computation a great deal

Model Formulation :

We shall present a "balance" formulation for the "Processing" Network .

The cumulative output of aggregate i is the sum of transferred output and inventory :

$$U_i(t) = \sum_{j \epsilon J(i)} V_{ij}(t) + I_i(t)$$

Intensity of aggregate j :

$$z_j(t) \le \min \begin{cases} \dfrac{NS_j}{D_j} \\ Z_j^P(t) - Z_j(t-1) \end{cases}$$

Output of aggregate j :

$$U_j(t) = Z_j^P(t+1-D_j)$$

Where the number of starts is :

$$Z_j^P(t) = \max \begin{cases} \lceil z_j(t) \cdot D_j \rceil + U_j(t-1) \\ Z_j^P(t-1) \end{cases}$$

and :

$$Z_j^P(t) \le \sum_i V_{ij}(t-1) \cdot W_{ji} \cdot \frac{n_j}{n_{ij}}$$

Output monotony :

$V_{ij}(t) \ge V_{ij}(t-1)$ **,for each** ij .

Capacity constraints :

$$\sum_i a_{ik} \cdot z_i(t) \le X_k(t) \quad k = 1,\dots,K$$

5.2.4 An inclusive model for Processing and Project Networks.

As shown in a previous section the accurate modeling of Aggregate *Project Networks* requires the examination of the level of support for *each sub-aggregate* . This requirement makes the modeling relatively complicated and decreases the attractiveness of the aggregation . We shall propose an approximate model , based on the *Aggregate Processing* model. This approximation replaces the *minimum* of distinct inputs of each sub-aggregate, by a *weighted sum* of the same inputs.*.

The "requirement" from the summation is that the weighted sum of the *final* inputs to the aggregate will equal the number of "transfers" which should be produced by the aggregate. This number is obviously the number of detailed activities which are included in the aggregate. The detailed construction of the "compensation" factor γ_{ij}, is described in appendix A. If we to maintain a balanced "feeding" procedure which provides similar level of support by *all the preceding aggregates*, we minimize the lost accuracy while we regain single-level relationships among the aggregates .

Proposed model :

The cumulative output of aggregate i is :

$$U_i(t) = \sum_{j \in J(i)} \beta_{ij} V_{ij}(t) + I_i(t)$$

Intensity of aggregate j :

$$z_j(t) \le \min \begin{cases} \dfrac{NS_j}{D_j} \\ Z_j^P(t) - Z_j(t-1) \end{cases}$$

* The use of weighted sums of inputs to sub-aggregates also appears in the Leachman-Boysen model. See Boysen [1982] and Hackman [1983].

Output of aggregate j :

$$U_j(t) = Z_j^P(t+1 - D_j)$$

Where the number of starts is :

$$Z_j^P(t) = \max \begin{cases} \lceil z_j(t) \cdot D_j \rceil + U_j(t-1) \\ Z_j^P(t-1) \end{cases}$$

and :

$$Z_j^P(t) \le \sum_i V_{ij}(t-1) \cdot W_{ji} \cdot \frac{n_j}{n_{ij}} \cdot \gamma_{ij}$$

(Each unit of input is multiplied by the appropriate "explosion factor", the weight of the receiving sub-aggregate and the "compensation" factor).

Output monotony :

$V_{ij}(t) \ge V_{ij}(t-1)$,for each ij .

Capacity constraints :

$$\sum_i a_{ik} \cdot z_i(t) \le X_k(t) \quad k = 1,\dots,K$$

The reader should note that by setting γ_{ij} and β_{ij} to equal 1, we return to the model which was derived for Processing Networks. Thus, by a careful usage of this model we can represent combinations of *shared* and *distinct* inputs and outputs.

Computational Simplifications :

Some of the ideas which were introduced in section 5.1 are valid for any aggregation of activities :

a. Given the due-date of the project, each aggregate can operate only within a period of time which is bounded between the Early Start and the Late Finish of the aggregate. These points of time are invariant with the *actual schedule* that we select.

b. The intensity and the transfers of the aggregate are constant at times external to this period .

These observations enable us to restrict our attention to time points at which the aggregates can be active and eliminate the constraints for the irrelevant time points . We can further decrease the size of the problem if we choose to partition the "range of operation" of each aggregate, into size-dependent intervals at which we shall examine the "balance" of transfers.

6. Model Validation.

6.1 Testing Procedure.

The purpose of the testing is to demonstrate the applicability of the proposed aggregate model for the development of meaningful aggregate measures for production networks. We shall demonstrate examples of Processing, as well as Project Networks. Both examples use activities which release output only upon completion of the task.

The model as described in chapter 5 enables us to use a Linear Program, but due to practical limitations (such as integer variables) we prefer to obtain our results using heuristic simulations. The "quality" of the model will be evaluated by comparing the aggregate resource utilization as produced by simulations of the aggregate models to the ones obtained directly from the underlying detailed networks. The detailed and aggregate models attempt to achieve similar objectives. The loading policy that we shall use for both systems will be "Earliest Loading". The heuristic "looking ahead" procedure that we shall use sets relative priorities based on the "slack" of the detailed or aggregate activities. Through the following discussion we shall use *LS* and *LF* for late start and finish accordingly.

The slack of a detailed activity is defined as : $S_i(t) = LS_i - t$

An underlying assumption is that the duration of each detailed activity is predetermined and is *not* a decision variable of the simulation. It is important to note that the only decision that we ought to take concerns the starting time of the activity. Systems with flexible intensity (and thus duration) were shown to be useful in the *disaggregation phase* (see Dincerler [1984]) but for planning purposes we prefer the traditional CPM approach which uses predetermined estimates for the duration of activities.

The definition of the slack in the aggregate system is much more complicated. We attempt to define a measure which is equivalent to the detailed slack and utilizes the *aggregate* network parameters :

Let A denote an aggregate while i denotes a detailed activity, then by definition :

$$LS_A = \min \left\{ LS(i) \mid i \in A \right\}$$

$$LF_A = \max \left\{ LF(i) \mid i \in A \right\}$$

The remaining working time on aggregate A , assuming an uncapacitated network, can be estimated by :

$$[\, LF_A - LS_A \,] \cdot \frac{n_A - Z_A(t)}{n_A}$$

Where n_A is the number of units that should be produced by A (which is the number of included detailed activities), and $Z_A(t)$ is the amount of work invested in A by time t measured in units of processed transfers.

The aggregate slack will be defined as :

$$S_A(t) = LF_A - [\, LF_A - LS_A \,] \cdot \frac{n_A - Z_A(t)}{n_A} - t \qquad (6.1)$$

The conversion to aggregate networks introduces an additional difficulty which is the *change of structure* when we transform the detailed network into the aggregate one. An additional decision is required, regarding the allocation of the output of each aggregate among its followers. In the detailed Project Networks, this additional decision did not exist due to the assumption of "distinct outputs". In Processing Networks the decision is transferred from the level of detailed activities to the level of aggregates.

6.1.1 Output Allocation for Production Networks.

In the allocation of the outputs , we shall use the following decision rule which attempts to equalize the lateness of the followers by supporting the followers which are behind, as measured by the slack :

The computation of the *target allocation* ΔV_{AB}, which is the incremental transfer from aggregate A to aggregate B, will be based on the definition of the *Potential Slack* which is *the slack realized if all the inputs at time t , denoted by $IN_B(t)$, were processed* . $IN_B(t)$ is a short notation for the weighted sum of inputs of aggregate B by time t and this expression is derived for both Project and Processing networks in chapter 5 as an upper bound on the "starts".

Under this definition the Potential Slack of aggregate B can be written as :

$$PS_B(t) = LF_B - (LF_B - LS_B)\cdot\frac{n_B - IN_B(t)}{n_B} - t$$

Or after simplification :

$$PS_B(t) = LS_B + (LF_B - LS_B)\cdot\frac{IN_B(t)}{n_B} - t$$

For notational convenience, we define $\bar{S}_B(t) = PS_B(t) + t$. Let us attempt to equalize the Potential Slacks of the followers at $t+1$, i.e., $\bar{S}_B(t+1) = \bar{S}$ for each B which follows A.

We can write $\bar{S}$ in the following form :

$$\bar{S} = LS_B + \frac{(LF_B - LS_B)}{n_B}\cdot(IN_B(t) + \Delta IN_B) = \bar{S}_B(t) + \Delta IN_B\cdot\frac{LF_B - LS_B}{n_B}$$

The incremental contribution of aggregate A to aggregate B can be written as :

$$\Delta IN_{AB} = \frac{\bar{S} - \bar{S}_B(t)}{LF_B - LS_B}\cdot n_B \qquad (6.2)$$

We consider two alternative policies to relate the input increment ΔIN_{AB} to the incremental output which is delivered to B from A:

a) Equalizing the slacks of the followers of A at $t+1$ considering only the contribution of aggregate A .

Under this approach, $\Delta IN_{AB} = \Delta V_{AB} \cdot \gamma_{AB} W_{AB} \cdot \frac{n_B}{n_{AB}}$ (See appendix A).

b) Equalizing the slacks of the followers of A at $t+1$, assuming similar "support" from the rest of the predecessors of each B . Each aggregate B may have several predecessors where A is one of them. By similar "support" we mean that the inputs arrive with a *constant mix*, proportional to $\frac{1}{\gamma_{AB} W_{BA}}$ for each source of input.

Under this approach $\Delta IN_{AB} = \Delta V_{AB} \cdot \frac{n_B}{n_{AB}}$ where an output unit of the predecessor is converted into an input unit by the appropriate "explosion factor" .

For reasons discussed earlier (model approximation) the second approach is more desirable and the derivation hereafter will use this approach.

If we define INV_A as the number of transfers available at A, then the weighted sum of the allocations to the followers must not exceed this inventory. Our policy is to allocate the whole inventory, thus :

$$INV_A = \sum_B \Delta V_{AB} \cdot \beta_{AB} = \sum_B \Delta IN_{AB} \cdot \frac{n_{AB}}{n_B} \cdot \beta_{AB} \qquad (6.3)$$

After substituting equation 6.2 into 6.3 and some algebraic manipulations we can determine what $\bar{S}$ should be :

$$\bar{S} = \frac{INV_A + \sum_B \left[\bar{S}_B(t) \cdot \frac{n_{AB} \cdot \beta_{AB}}{(LF_B - LS_B)} \right]}{\sum_B \frac{n_{AB} \cdot \beta_{AB}}{(LF_B - LS_B)}}$$

Using this result and the result for ΔV_{AB} we get the target allocation :

$$\Delta V_{AB} = \frac{\bar{S} - \bar{S}_B(t)}{LF_B - LS_B} \cdot n_{AB} \text{ , subject to : } V_{AB} \leq n_{AB}$$

6.1.2 Structural Constraints :

Processing networks have "additive" inputs. In these networks we restrict our attention to the cumulative contributions of the inputs, regardless of their origin.

The situation in "project networks" is more complicated. Besides the conservation of transfers, we ought to consider the special (logical) "AND" nature of the inputs of the detailed activities. An immediate consequence of the "Early Schedule" policy is a "push" material-flow system. Since the aggregates have an additive input structure, we have to provide a mechanism to secure the "CPM" structure.

A careful analysis of our treatment of Project Networks (see 5.2.3) reveals that we transform the network into an "equivalent" Processing network by using several weighting factors. Due to this conversion we are unable to monitor the "distinct" inputs and outputs directly. We attempt to maintain "balanced" inputs by restricting the weighted sum of inputs which may be "pulled" by an aggregate, to its *expected* value as described below.

We shall regard the arrival of inputs to each aggregate as a Renewal Process. If the amount of transfers is large enough we can expect, based on the "strong law of large numbers of Renewal Processes", that the expected number of arrivals will be proportional to the portion of the "input window" which has elapsed. The earliest "input window" starts at ES and ends at $EF-\bar{D}$, where $\bar{D}$ is the average duration of the included activities. We shall enforce the following bound on the input of aggregate B at time t:

$$IN_B(t) \leq N0_B + \frac{t-ES_B}{(EF_B-\bar{D}_B) - ES_B} \cdot (n_B - N0_B)$$

Where $N0_B$ is the number of unpreceded detailed activities in B.

The reader should note that this result is valid whether we deal with capacitated or with uncapacitated networks.

6.2 Simulation Techniques :

The detailed Project Network is most efficiently simulated using the "Discrete Events" technique. (Moving in time from one event to the next one. For detailed discussion see Law & Kelton [1982]). We can take advantage of the efficiency of this method due to the fact that no events are generated by an activity between its start and its end.

The aggregate network will be simulated using a discrete time grid. The "balance" of transfers and resource availability is done at each grid point. The previous method is not efficient here because the aggregate equivalent of the detailed network is composed of aggregates of the "processing" type and transfers are no longer easily defined as discrete events.

6.3 Cases Studied :

We constructed two basic networks for the testing of the models, a "Project Network" and a "Processing Network".

6.3.1 Project Network .

We selected a synthetic detailed network with the following characteristics :

a) 999 detailed activities which belong to 19 aggregates.

b) The flow of the network is always "upstream" (increasing indices of the aggregates).

c) Random selection of the followers. The selection is done in two stages:

1) The following aggregate is selected randomly. Let the current aggregate be indexed i and the highest ordered aggregate $i+k$. Then the probability to select aggregate $i+l$ is set to $\zeta \cdot \frac{2(k-l)}{k(k+1)}$: $l=1,\dots,k$,

and $1-\zeta$ is the probability to have a null arc. We made 10 random "tosses" for each node and set ζ to be $\frac{\delta}{10}$. Thus in average we have δ arcs leaving each node. We selected δ as 1.5 which we believe is representative of many practical networks.

2) In each aggregate, the specific detailed follower was selected randomly with equal probability to each detailed activity.

d) The number of detailed activities in each aggregate was selected such that there are fewer activities in the first and last aggregates and more toward the central ones. This structure was motivated by the following observation :

If we consider the aggregates as representing ordered production processes, we can expect that there are few activities which start from raw-materials, produce many types of intermediate products and converge again to a smaller number of activities (assembly) which deal with final products. As a consequence of this analysis we generated a Project Network of the following structure :

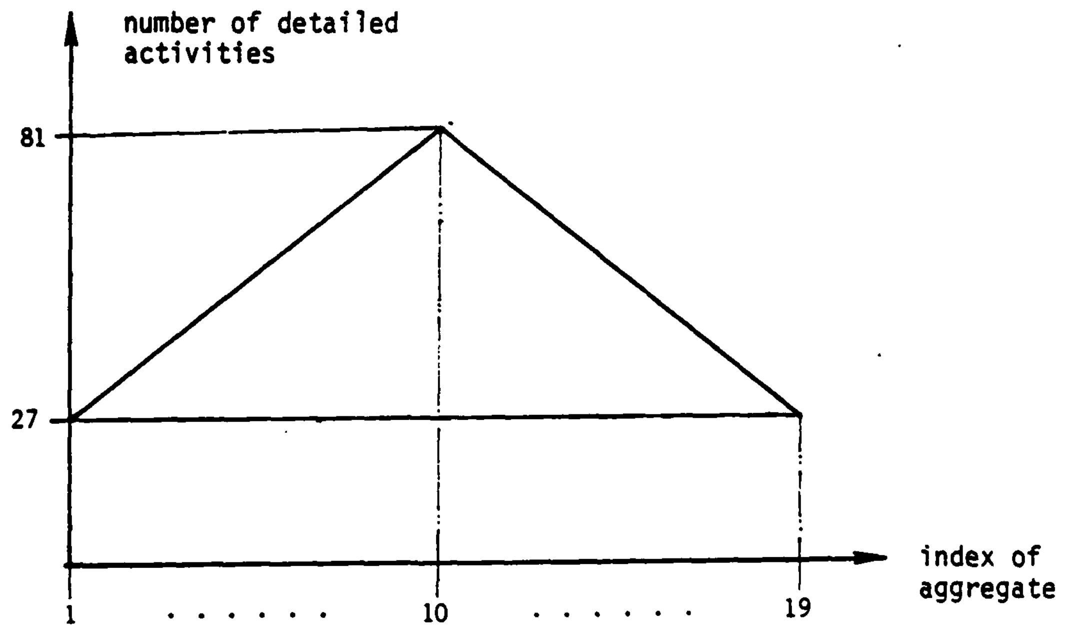


fig 6.3.1 : Number of activities in the processes (Project Network).

6.3.2 Processing Networks.

In the construction of the Processing Network we have used the same 19 aggregates (processes) as described for the Project Network.

The network consisted of 100 completely parallel "strings", each of which consisted of an ordered subset of the 19 processes. We partitioned the network into an "active" set of nodes and the set of nodes which were omitted. The active subset in each string was selected randomly in the following way :

Starting from process 1 :

```
1       for current process $i$ , and some aggregate $i+l$ the probability of connection is :

        $P(i+l) = density \cdot \frac{2(k-l)}{k(k+1)} : l=1,\dots,k$

        If the connection is null (The selected "node" is beyond $i+k$), we proceed as follows :
            * Remove activity $i$ from the "active" set.
            * $i = i+1$
            * Go to 1
        Else : (connection to $i+l$)
            * Remove activities between process $i$ and process $i+l$
            * $i = i+l$
            * Go to 1
```

The resulting network had 100 strings, 399 activities and 299 arcs. (We chose density = .75). The number of activities within the aggregates ranged from 8 to 80 (In aggregate 1).

6.4 Model Evaluation

Before we evaluate the the results of the tests that we made, we would like to restate the objectives of our tests.

Our model proposes a correction and generalization for the DLAAM. We use this model as an approximation to the transfer relationships between aggregates which consist of activities which require the same process. Our research was motivated by practical problems in which the traditional policy was of "Early Schedule". In order to complete the formulation of the model for *aggregate networks* we had to add three rules concerning the following :

a) The assignment of the "inventory" of each aggregate to its followers. (We denote this as the "push" rule).

b) Limiting the inputs of each aggregate at any time. This rule attempts to prevent irreversible excessive "pulling" of inputs by a certain aggregate on the account of another and thus "twist" the structure of the network. We term the integrated effect of these rules as the "Push-Pull" rules.

c) Allocation of resources, as well as inputs, is based on the "aggregate slack", which by itself is based on a linear approximation of the progress of the aggregates.

6.4.1 Evaluation of the Results for Project Networks.

Investigation of the Project Network which was introduced in section 6.3.1 shows that in spite of all the additional assumptions that we made, the model is appropriate for the representation of *uncapacitated* networks of this type. Figures 6.4.1-a and 6.4.1-b demonstrate the usefulness of the model for aggregate study of a network. Some of the resource-usage comparisons were not that good. It was clearly related to aggregation in which the number of included activities was too small to justify the "averaging" behavior of the aggregate model. The early peak shown in these figures is due to the "early start" loading policy. When we tried to model *capacitated* loading of the same network we got less satisfactory results. Though the completion of the project could be estimated with sufficient accuracy (the aggregate model gave estimates within -10% of the detailed one), the utilization of resources was represented correctly only for capacitated resources. If the "straight forward" approach does not satisfy the needed accuracy, one can use the model to generate a library of aggregate capacitated profiles of Project Networks where the start and finish parameters are taken from *capacitated* detailed networks. This "library" can serve as a tool in a multiple scenario analysis of multi-project decisions. Another idea which might solve the practical problem of capacitated Project Networks is to update dynamically the target dates for the aggregates in the "remaining network".

6.4.2 Applicability to Processing Networks.

Processing Networks proved to be less sensitive to erroneous allocation of the outputs of an aggregate to its followers. This result was expected due to the *additive nature of the inputs in the detailed level.* As a consequence, the model gives satisfactory results for both **capacitated as well as uncapacitated** profiles of resource usage. We demonstrate typical results for the same resources at three cases :

a) Uncapacitated simulations. (Figures 6.4.2-a and b).

b) Uniform capacity levels were set to 20% of the maximal level required from any resource in the uncapacitated run. (Figures 6.4.2-c and d).

c) Same. Capacity levels are set to 10% . (Figures 6.4.2-e and f).

In order to study the reasons for the (acceptable) discrepancies between the detailed and aggregate model we compared the curves of "Detailed activities ended" Vs. time, from the detailed and the aggregate models. The data for the Detailed model was complete. For the aggregate model our program provided only cumulative output for each aggregate at intervals of 10 days. Further investigation shows clearly that the discrepancy resulted mainly from the "Push-Pull " decision rule which controls the allocation of outputs among the followers of each aggregate. Typical results are introduced in figures 6.4.2-g and h . (obtained for the 20% capacity levels).

Another related observation was that in all the comparisons of "activities ended", the *aggregate* output curve was hardly ever ahead of the *detailed* finish curve. This result is desirable since we prefer an approximation which is slightly conservative over an "optimistic" and infeasible one. This result is consistent with the assumption that the "quality" of the decisions of the *detailed* system is superior to decisions based on *aggregate information*

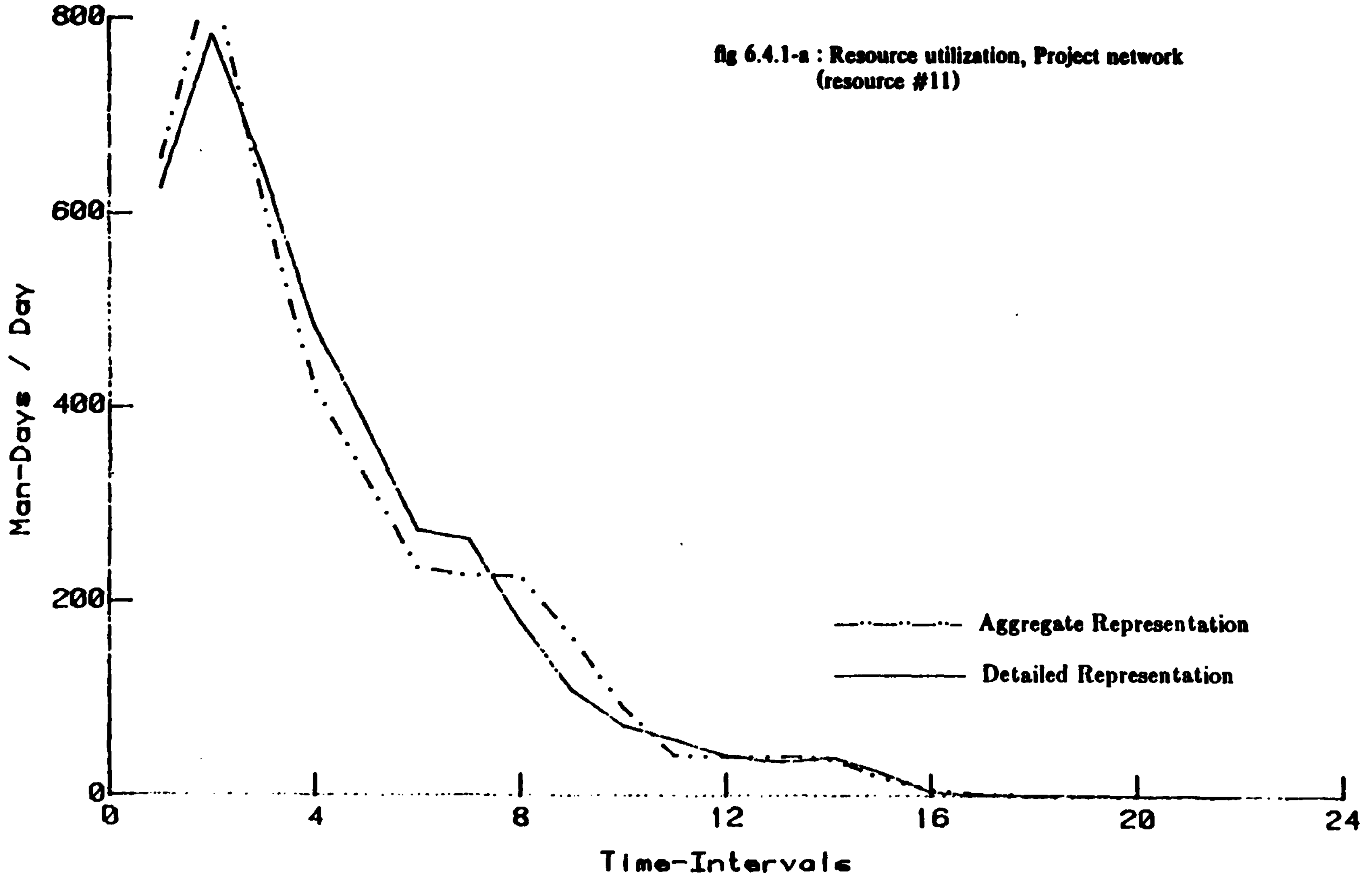


fig 6.4.1-a : Resource utilization, Project network
(resource #11)

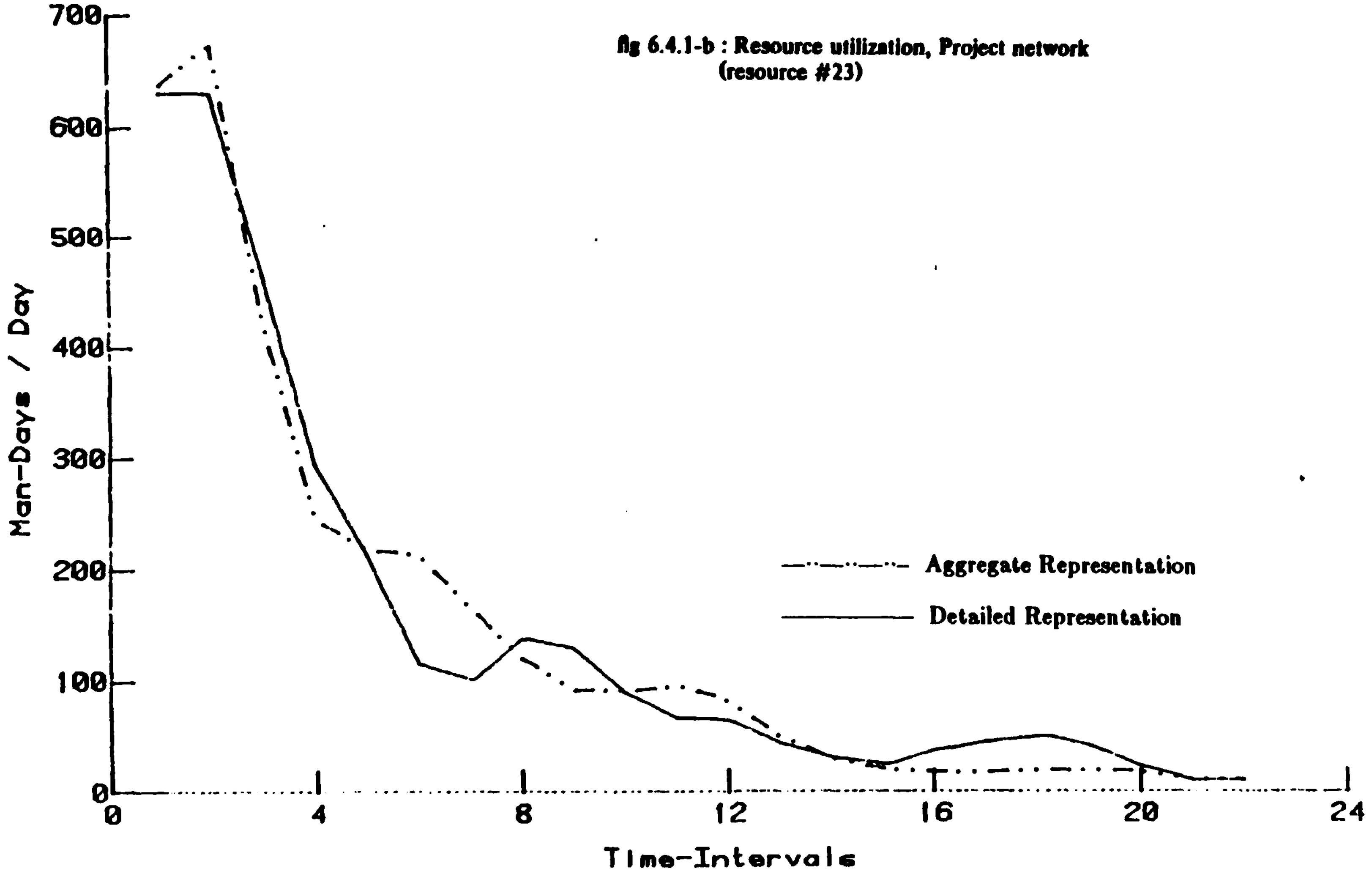


fig 6.4.1-b : Resource utilization, Project network (resource #23)

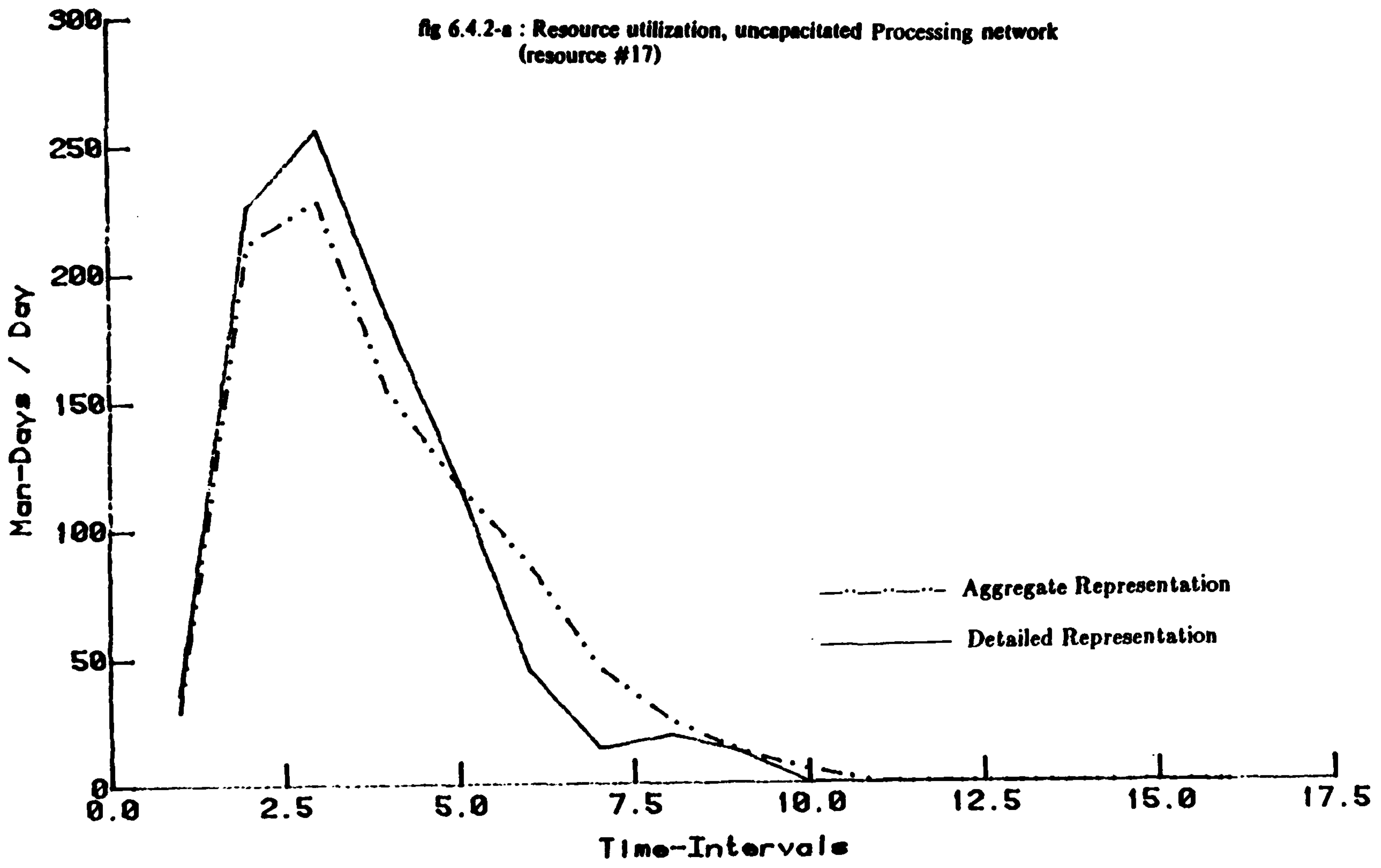


fig 6.4.2-a : Resource utilization, uncapacitated Processing network (resource #17)

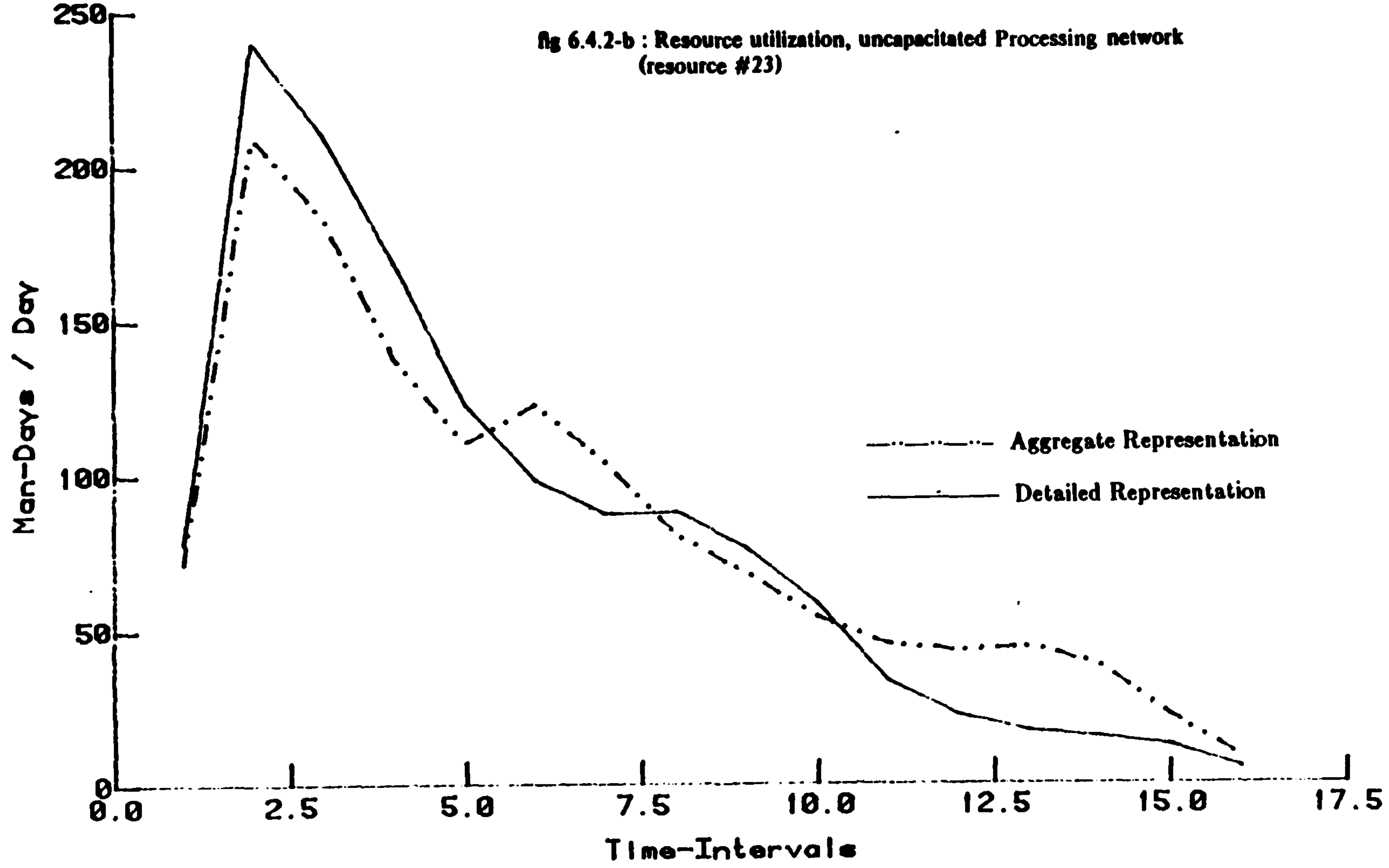


fig 6.4.2-b : Resource utilization, uncapacitated Processing network (resource #23)

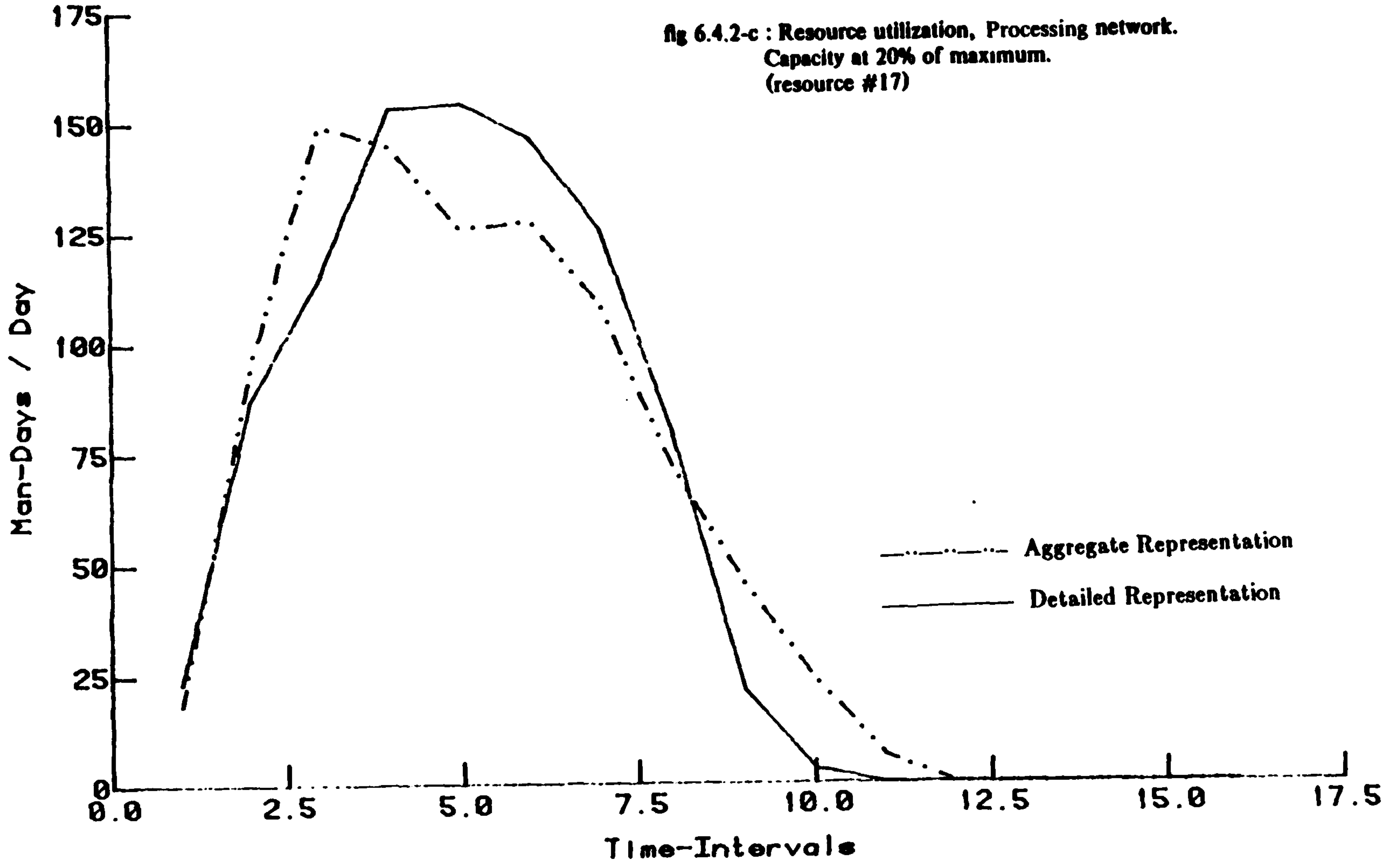


fig 6.4.2-c : Resource utilization, Processing network.
Capacity at 20% of maximum.
(resource #17)

fig 6.4.2-d : Resource utilization, Processing network.
Capacity at 20% of maximum.
(resource #23)

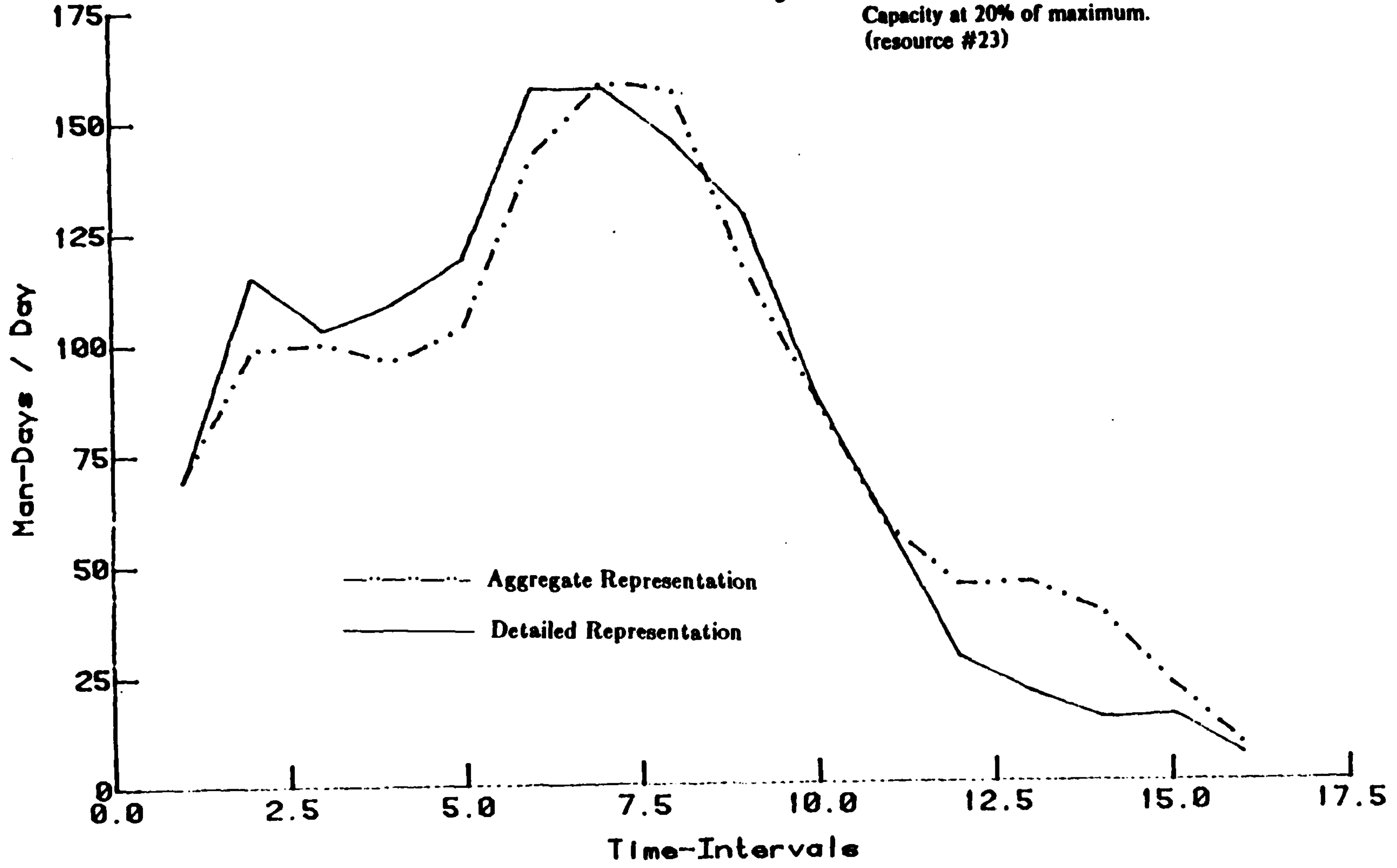
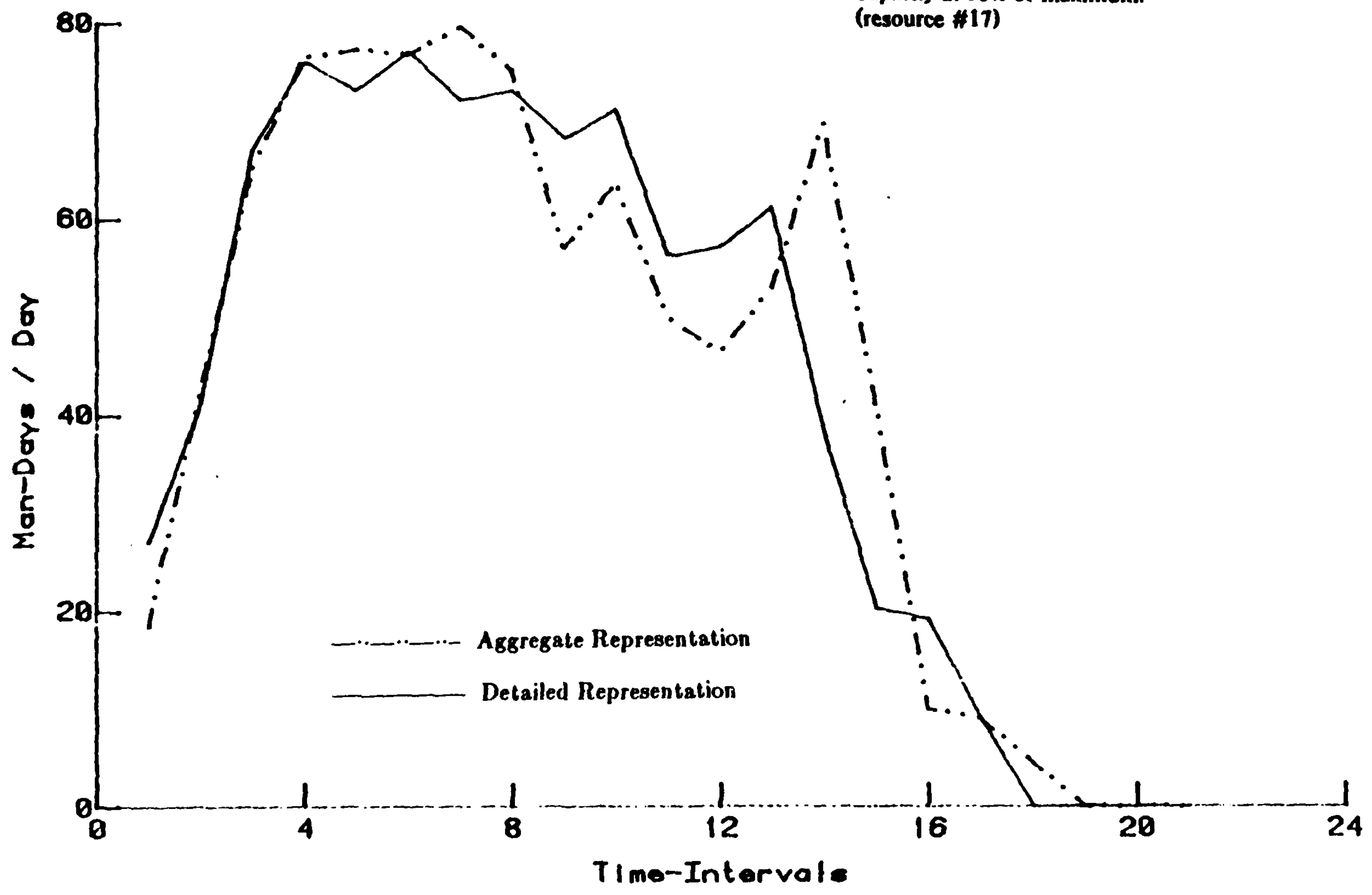


fig 6.4.2-e : Resource utilization, Processing network.
Capacity at 10% of maximum.
(resource #17)



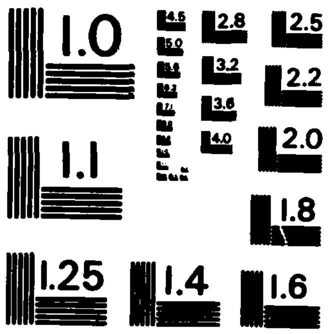
AD-A160 041 DYNAMIC MODELS OF DETAILED AND AGGREGATE PRODUCTION NETWORKS(U) CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER M MIZRACH SEP 85 ORC-85-10 2/2

UNCLASSIFIED N00014-76-C-0134 F/G 5/1 NL

END

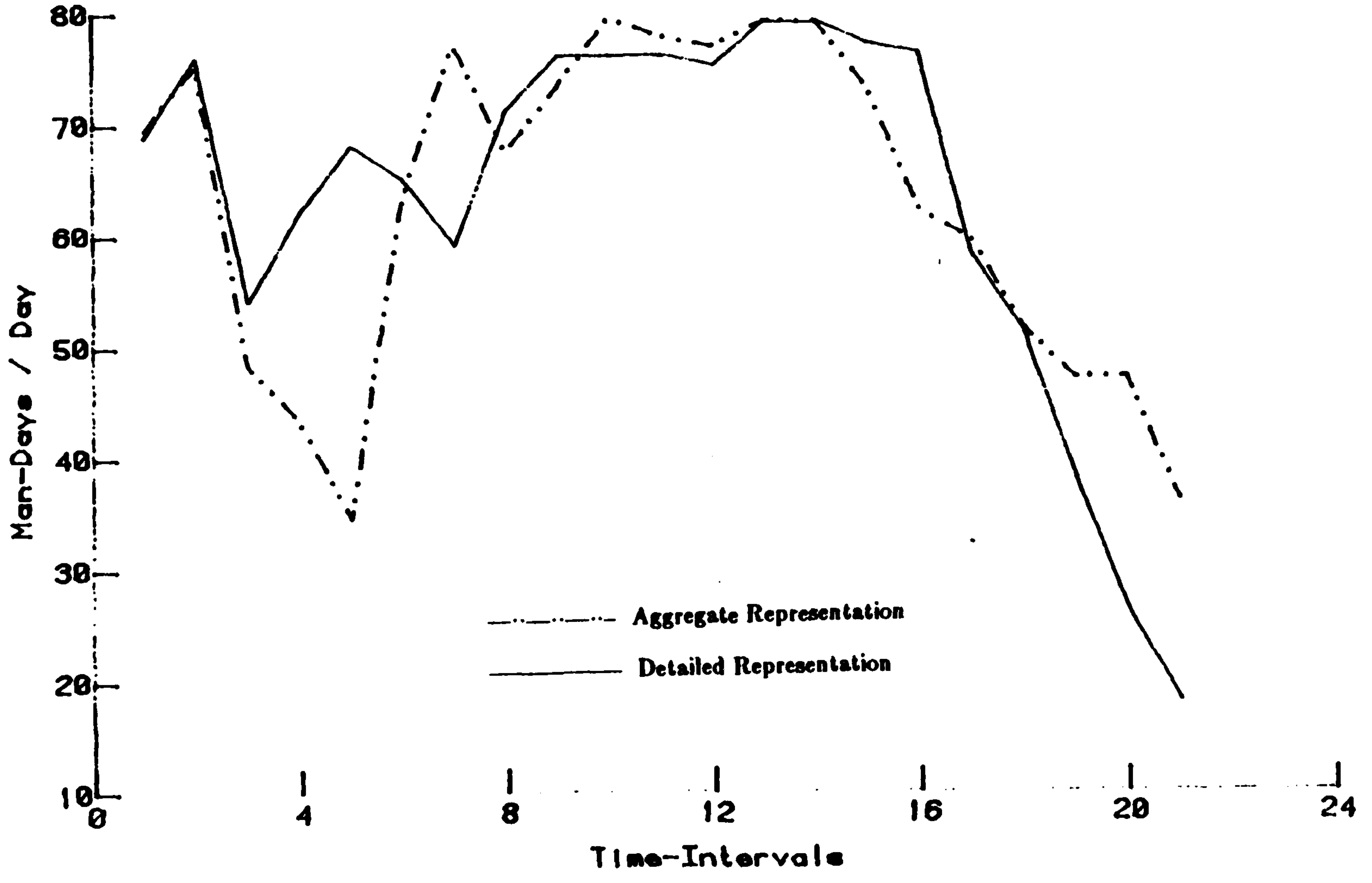
FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

fig 6.4.2-f : Resource utilization, Processing network.
Capacity at 10% of maximum.
(resource #23)



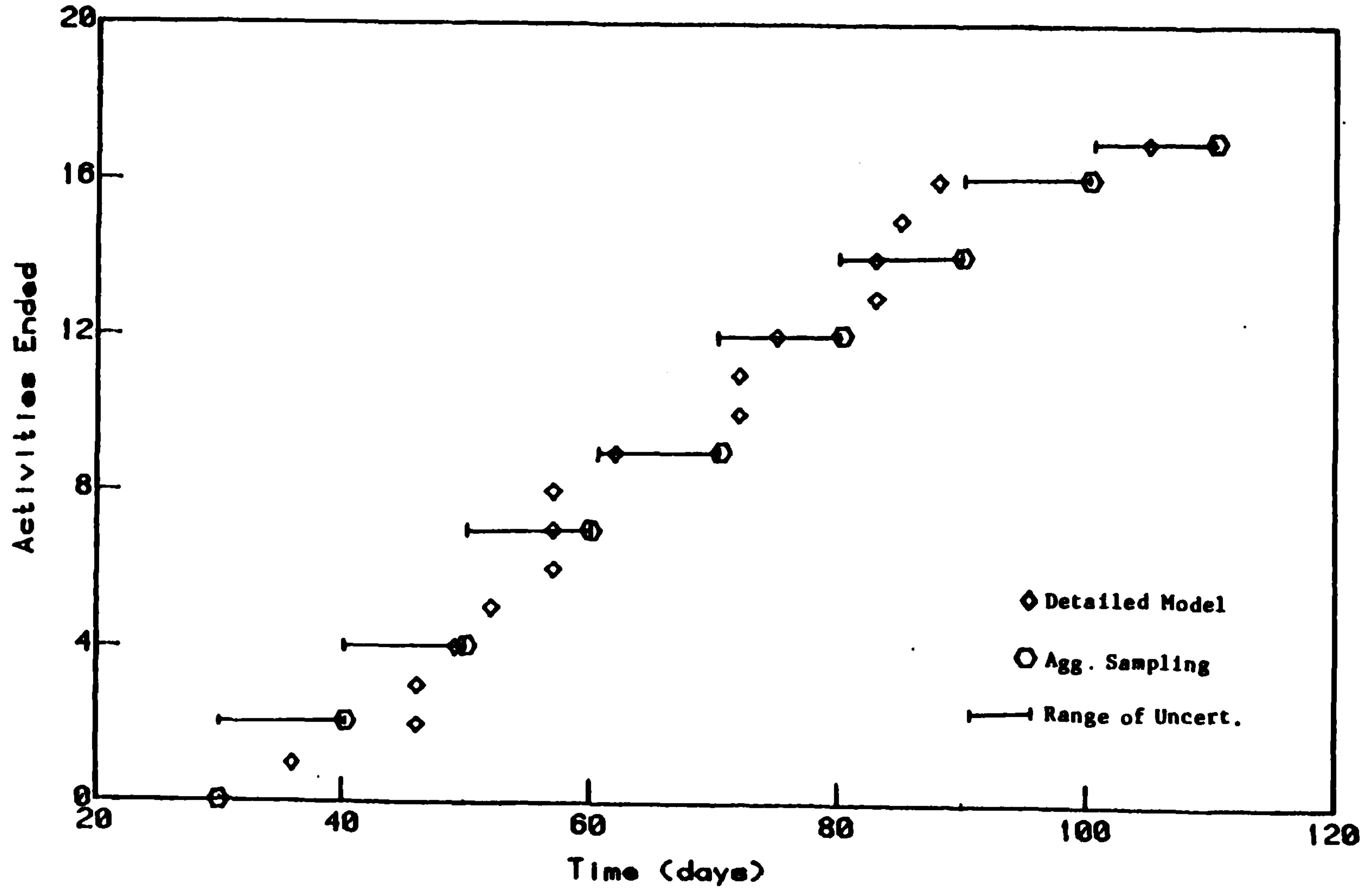


fig 6.4.2-g : Activities ended, aggregate #9

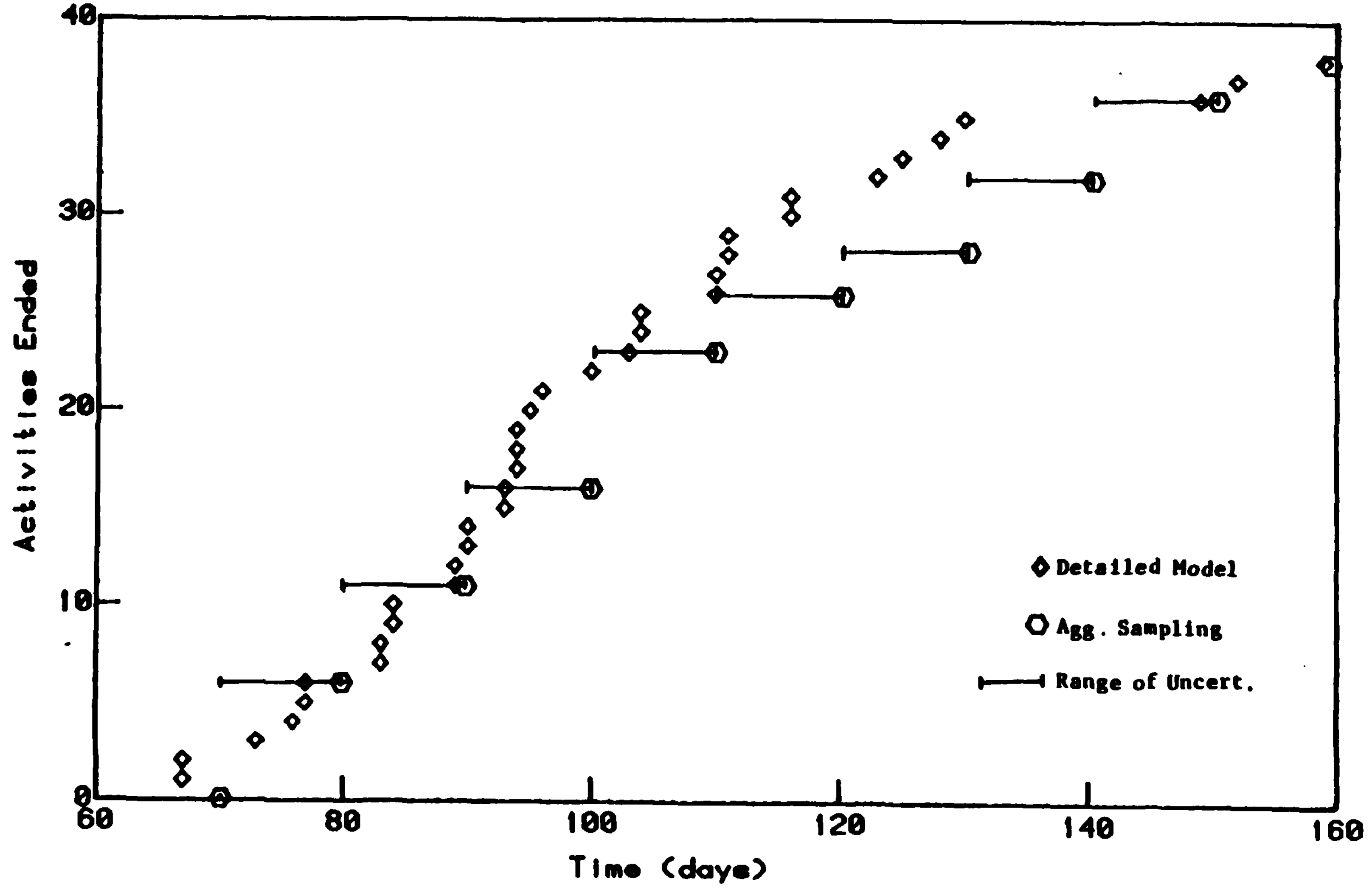


fig 6.4.2-h : Activities ended, aggregate #19

7. Conclusion

This research proposed a generalization to the Dynamic Linear Activity Analysis Model. The generalization involves the effects caused by the processing time of each transfer unit within the activities. This model is required whenever the production process is not instantaneous. In this case our model shows that it is necessary to distinguish between the amount of work which was invested in the activity and its actual output.

Later on we develop an analogy between the transfer relationships among activities of the Detailed Network and the transfer of *detailed activities* among aggregates, where the aggregation is based on similar mix of resources within the aggregate. The model was validated through simulations of the Detailed Network and its Aggregate representation. The main results are presented here :

Aggregate representation of Processing Networks were satisfactory for the prediction of the load on resources when the resources were either unconstrained or heavily capacitated.

Aggregate representation of Project Networks were satisfactory only when the resources were unconstrained. For the capacitated problems, the estimation of completion-times was sufficiently good but the utilization of the resources which did not operate at their capacity, as represented by the aggregate model, was not close to the detailed resource utilization. This limitation can be cured by using capacitated Early Start and Finish which can be obtained from constrained simulation of the Detailed Network. The merit of the last approach is in *aggregate representation of capacitated projects* in a Multi-Project environments. (In the example that we tested, the critical path had no slack. It can be expected that in less constrained problems where an initial slack exists the CPM network may remain feasible even under capacity limitations. Thus the aggregate representation will be appropriate for capacitated problems as well).

Further Research.

There are several main areas at which we hope that further research might improve the accuracy and applicability of our models :

1.The transfer relationships which were developed for the aggregate project networks are merely an approximation which represents the project network by a weighted processing network. This approximation has the following two shortcomings :

a) The "correction" factors are constructed such that the sum over the final inputs or outputs of an aggregate will equal the number of detailed activities which are included in the aggregate. This sum is not necessarily correct during the "active" period of the aggregate.

b) The construction of the correction factors is based on the limiting assumption of close similarity between the activities which belong to the same aggregate.

We hope that these shortcomings will be resolved by new ideas proposed by future researchers.

2) We would like to see more parametric study which will investigate the possible relaxation of the similarity requirement thus expand the range of practical problems which can be solved properly by these models. We also recommend the investigation of other approaches for dealing with non-identical activities which belong to the same process . One such approach might be to define a typical transfer batch and assign multiple counts to larger activities.

3) The *processing network* that we have tested is typical for manufacturing facilities in which the process is unique or predetermined. In these production "networks" the representation of independent strings is appropriate. We have not *tested* the case where we consider alternative sources and destinations. Though we do not expect any con-

ceptual difficulty in using our model for these networks, the simulation tools that we developed assume a predetermined task for each detailed activity. In our opinion, the simulation of moregeneral networks will also require *different loading policies* such as Latest Loading.

4) Both models, the detailed and the aggregate DLAAMP are based on restrictive assumptions concerning the allocation of "service" resources to the transfer units which entered the activity. We hope that the ideas introduced in this thesis will contribute to the development of similar models for environments which require different sets of assumptions.

5) In our Model Validation we emphasized the testing of the appropriateness of the model of transfer conservation. The problems that we chose to solve are related to the "makespan" of a project or a group of similar processing jobs. We hope that this model will serve as an important building-block in the solution of various practical problems in the area of Production Planning.

Appendix A

The treatment of aggregate which consist of detailed activities with complementary inputs, distinct outputs, or both, is quite complicated. In this appendix, we shall propose an approach which attempts to simplify the relationships among the aggregates in the cost of some approximation. The notation which we use in this appendixed was defined in chapter 5.

1. Simplifying the output relationships :

Our assumption for distinct outputs of an activity is that they are produced simultaneously. An example might be, dies of different quality which are "members" of the same wafer in the production of integrated circuits.

We shall further assume that these distinct outputs are *allocated simultaneously* to their distinct "consumers". If we assume that the distinct outputs are produced within a known proportion, we regard a transfer unit of this proportion as our output unit. In order to obviate multiple-counting of the same output while we model the aggregate relationships, we would like to weigh the outputs such that the summation over the outputs at any time will be correct. Unfortunately we consider this approach unrealistic and "compromise" on the following requirement for aggregate i :

$$\sum_{j} V_{ij}(T)\cdot\beta_{ij} = n_i$$

In words, the final contributions of the weighted outputs of aggregate i equals its "target" output which in our system is the completion of all the included detailed activities. β_{ij} is the weighting factor for the output of i which is delivered to j .

Let m_l be the number of *aggregates* which receive output from activity l which belongs to aggregate i , and let $y(l,j)$ be an indexing function such that :

$$y(l,j) = \begin{cases} 1 & \text{if } l \to j \\ 0 & \textit{otherwise} \end{cases}$$

We define β_{ij} as follows :

$$\beta_{ij} = \frac{1}{n_{ij}} \sum_{l} \frac{y(l,j)}{m_l}$$

The rationale for this expression is that we sum over the "portions" of activities $l : l \epsilon i$, which "feed" aggregate j . We claim that the construction of β_{ij} conforms with the previously stated requirement. The proof is as follows :

$$V_{ij}(T) = n_{ij}$$

$$\sum_{j} \beta_{ij} \cdot n_{ij} = \sum_{j} \sum_{l} \frac{y(l,j)}{m_l}$$

By rearranging the summation :

$$\sum_{j} \beta_{ij} \cdot n_{ij} = \sum_{l} \sum_{j} \frac{y(l,j)}{m_l}$$

But $\sum_{j} \frac{y(l,j)}{m_l} = 1$ by construction and

$$\sum_{j} \beta_{ij} \cdot n_{ij} = n_i$$

2. Simplifying the input relationships.

We use a similar approach to simplify the input relationships. Suppose that we have some way to "feed" each detailed activity $l : l \epsilon j$ in a "balanced" manner such that the mix of its inputs is kept. A unit of this mix should be counted only as a single unit of input. We approximate this relationship by the weighing of the transfers among the aggregates. As the reader might recall, The input V_{ij} should be multiplied by some

additional weighting factors :

1. The "explosion factor" $\frac{n_j}{n_{ij}}$.

2. The relative weight of the part of aggregate j which receives input from aggregate i , W_{ji} .

Under our assumption of "similar activities", $W_{ji} = \frac{n_{ji}}{n_j}$, thus the product of the two weighting factors gives $\frac{n_{ji}}{n_{ij}}$

Let m_l be the number of aggregates which "feed" activity l in aggregate j , and

$$y(i,l) = \begin{cases} 1 & \text{if } i \rightarrow l \\ 0 & otherwise \end{cases}$$

We define the "summation weighting factor" γ_{ij} as follows :

$$\gamma_{ij} = \frac{1}{n_{ji}} \cdot \sum_l \frac{y(i,l)}{m_l}$$

It is easy to show, using the same technique as in the previous section, that the weighted sum of the final inputs adds to n_j , as it should.

Appendix B : Additional assumptions required for DLAAMP

This appendix attempts to explain assumtions 6 through 8 as stated in section 2.4 .

DLAAMP requires three assumptions in addition to those stated explicitly for DLAAM. All this assumptions are needed in order to establish a unique production function.

1) Availability of material inputs :

The discipline of application of "service" resources to transfer-units should enable the activity to operate in a way which satisfies the following requirements :

a) In the case of "complementary" inputs (i.e. assembly), the various types of inputs are allocated such that transfers are combined as fast as possible.

a) Each combined transfer is immediately available for processing.

FIFO is *one* of the appropriate disciplines. We selected it due to the simplicity of its application.

2) The next two assumptions (numbered as 7 and 8) are strictly related and will be discussed simultaneously.

In many production systems the time required for the processing of a unit is completely determined. Examples are most of the Chemical processes and "machine intensive" metal processing. Processes which involve a great deal of manual operation tend to have larger variance in processing-time, but for the purpose of scheduling we should consider a fixed representative processing-time per unit. The assumption of continuous processing at a predetermined (uniform) rate enables us to use a production function which depends only on the cumulative consumption of resources (materials and services). We shall demonstrate it by the following example and counter-example.

Suppose activity i has more than 2 servers. At the beginning of period 1 we have one unit ready for processing. Another unit arrives at the end of period 5 . The typical processing time is 10 days.

Under the previous assumptions :

$$z_i(t) = \begin{cases} 1/10 & t \leq 5 \\ 2/10 & 5 < t \leq 10 \\ 1/10 & 10 < t \leq 15 \end{cases}$$

$Z_i(5) = .5$; $Z_i(10) = 1.5$; $Z_i(15) = 2$

$Z_i^P(1) = 1$; $Z_i^P(6) = 2$

The output is uniquely defined by the production function :

$U_i(t) = Z_i^P(t+1 - \lceil D_i \rceil)$ such that $U_i(10)=1$ and $U_i(15)=2$.

The unique output results from the unique mapping between "starts" and the applied "service" resources.

Now, let's assume that the first unit is processed at the rate of 1/20 units per day. It is easy to verify that the state variables assume the following values :

$Z_i(5) = .25$; $Z_i(10) = 1.0$; $Z_i(15) = 1.75$

$Z_i^P(1) = 1$; $Z_i^P(6) = 2$

Even though the cumulative consumption of resources is sufficient to complete a transfer unit by time 10, evidently, no output is available at that time. As a conclusion, the production function is not well defined and the cumulative consumption of resources is not sufficient information to determine output when the previously specified assumptions are violated.

REFERENCES :

Boysen J.

(1982) "Aggregate Project Model for Resource Allocation Within Multiproject Construction Systems" , Unpublished Ph.D Thesis, University of California, Berkeley.

Cooper D.F.

(1976) "Heuristicfor Scheduling Resource Constrained Projects : An experimental Investigation." *Management Science,* 22 (11), 1186-1194.

Davis E. W.

(1973)"Project Scheduling under Resource Consraints - Historical Review and Categorization of Procedures. " *AIIE Transactions,* 21 (8), 944-955.

Dincerler A. I.

(1984) "Project Scheduling in Project - Oriented Production Systems. " Unpublished Ph.D Thesis, University of California-Berkeley.

Hackman S. T.

(1983) "A General Model of Production : Theory and Application." Unpublished Ph.D Thesis, University of California-Berkeley.

Hax A.C., Candea D.

(1984) *Production & Inventory Control,* Prentice Hall, Englewood Cliffs, N.J.

Heyman D.P. , Sobel M. J.

(1982) *Stochastic Models in Operation Research Vol 1.* , McGrow-Hill, New-York.

Holt C. C., Modigliani F., Muth J. F. and Simon H. A.

(1960) *Planning Production, Inventories and Work Force* , Prentice-Hall, Englewood Cliffs, N.J.

Johnson L. A. and Montgomery D. C.

(1974) *Operation Research in Production Planning, Scheduling, and Inventory Control*, John Wiley and Sons, New-York.

Kasper M.

(1983) "Production Planning of Capacitated, Multi-stage Discrete Manufacturing Systems", Unpublished Ph.D Thesis, University of California, Berkeley.

Koopmans T. C.

(1951) *Activity Analysis of Production and Allocation*, John Wiely & Sons, New-York.

Law A. M., Kelton W. D.

(1982) *Simulation Modeling and Analysis*, McGraw-Hill, New-York.

Leachman R. C.

(1982) "Production Planning for Multi-Resource Network Systems" *Naval Research Logistics Quarterly*, 29 (1) , 47-54

Leachman R. C.

(1983) "Production Planning for Multi-Resource Leveling in Construction Systems Through Variation of Activity Intensities" *Naval Research Logistics Quarterly*, 30 (3), 187-195

Leachman R. C. and Boysen J.

(1983) "Aggregate Network Model of Ship Overhaul," ORC Report 82-2, Operations Research Center, University of California, Berkeley.

Leontief W. W.

(1951) *The Structure of the American Economy 1900-1939*, Oxford University Press.

Mendelssohn R.

(1980) "Improved Bounds for Aggregated Linear Programs." *Operations Research* " 28 (6), 1450-1453.

Morgenstern O., ed.

(1954) *Economic Activity Analysis,* John Wiley $ Sons, New-York.

Patterson J. H.

(1973) "Alternate Methods of Project Scheduling with Limited Resources" *Naval Research Logistics Quarterly*", 20 (4), 767-784

Silver E.A., Peterson R. (1985) *Decision Systems for Inventory Management and Production Control* John Wiley & Sons, New-York.

Pritsker A. A. B., Waters L. S. and Wolfe P.M.

(1969) "Multi-Project Scheduling with Limited Resources : A Zero-One Programming Approach" *Management Science,* 16 (1), 93-108

Shepard R. W., Al-Ayat R. and Leachman R.C.

(1977) "Shipbuilding Production Function : An Example of a Dynamic Production Function" in *Quantitative Wirtschaftsforschung Festschrift Volume [Wilhelm Krelle],* J.B.C. Mohr (ed.), Tubingen, Germany

Talbot F. B. and Patterson J. H.,

(1978) "An efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems" *Management Science*, 24 (11), 167-168

Zipkin P.

(1980) "Bounds on the Effect of Aggregating Variables in Linear Programs" *Operations Research*, 28, 903-916.